

Convex Mathematical Programs for Relational Matching of Object Views

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl.-Phys.
Christian Schellewald
aus Hohenlimburg

Mannheim, 2004

Dekan:	Professor Dr. Matthias Krause, Universität Mannheim
Referent:	Professor Dr. Christoph Schnörr, Universität Mannheim
Korreferent:	Professor Dr. Joachim Denzler, Universität Jena
Tag der mündlichen Prüfung:	14. Juli 2005

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Overview	2
1.2.1	Graph Matching	2
1.2.2	Exact and Inexact Graph Matching	3
1.2.3	What this thesis is about	3
1.2.4	What this thesis is not about	4
1.3	Convex Optimization	4
1.3.1	Semidefinite Programming	5
1.4	Contribution	5
1.4.1	QAP Convex Relaxation used for Graph Matching	5
1.4.2	Subgraph Matching by Semidefinite Programming	6
1.5	Outline	6
1.6	Related Work	7
1.6.1	Basic Graph Matching Approaches	7
1.6.2	More General Graph Matching Approaches	11
2	Preliminaries	15
2.1	Basic Graph Theory	15
2.2	Graph Matching	16
2.2.1	Bipartite Matching	16
2.2.2	Representation of Matchings	16
2.2.3	Classification of Graph Matching Problems	18
2.3	Complexity	19
2.3.1	Complexity Classes	20
2.3.2	Complexity of Discussed Graph Matching Approaches	21
2.4	Terminology and Mathematical Preliminaries	21
2.4.1	Notation	21
2.4.2	Operator Definitions	22
2.5	Basic Convexity Concepts	23
2.5.1	Convex functions and sets	24
3	Combinatorial Relaxation and Convex Optimization	27
3.1	Relaxations and Lower Bounds	27
3.1.1	Interpretation of the Relaxation as Approximation	28
3.1.2	Lagrangian Relaxation	29

3.1.3	Convex Relaxations	30
3.2	Optimality Conditions	30
3.2.1	Lagrangian Duality	31
3.2.2	Karush-Kuhn-Tucker Conditions	31
3.3	Convex Optimization Problems	32
3.3.1	General Convex optimization problems	32
3.3.2	Convex optimization problems in this thesis	33
3.3.3	Linear Programming	33
3.3.4	Quadratic Programming	34
3.3.5	Semidefinite Programming	35
3.4	Solving Convex Optimization Problems	36
3.4.1	Interior Point Methods	36
3.4.2	Solver for Convex Optimization Problems	36
3.5	Computing Integer Solutions for Assignments	37
4	Weighted Graph Matching	41
4.1	Problem Statement	41
4.1.1	The Quadratic Assignment Problem	42
4.1.2	Graph Matching as QAP	42
4.2	Relaxations and Lower Bounds	45
4.2.1	Permutation Matrices and Relaxations	45
4.2.2	Orthogonal Relaxation	46
4.2.3	Projected Eigenvalue Bound	47
4.2.4	Convex Relaxation	48
4.2.5	Combinatorial Solutions	50
4.2.6	The 2opt Post-Processing Heuristics	51
4.3	Other Approaches	51
4.3.1	The Approach by Umeyama	51
4.3.2	Graduated Assignment	52
4.3.3	SDP Relaxation	53
4.4	Convex Relaxation: An Illustrative Numerical Example	54
4.4.1	A Small Graph Matching Problem	55
4.4.2	Relaxations and Bounds	55
4.4.3	Visualization	57
4.5	Implementation Details	58
4.6	Experiments	59
4.6.1	QAPLIB Benchmark Experiments	60
4.6.2	Random Ground-Truth Experiments	61
4.6.3	Real World Example	64
4.7	Discussion	66
4.7.1	Invariants	66
4.7.2	Limitations of the QAP Graph Matching	67
4.8	Towards Subgraph Matching	68
4.8.1	A Simple Extension Approach	68
4.8.2	A Promising Subgraph Matching Approach	69
4.8.3	Comparison of the two Approaches	71
4.8.4	Relaxation Attempt	72

4.8.5	Projection Approach	74
5	Subgraph Matching	75
5.1	Problem Statement	75
5.2	Notation and Bipartite Matching	76
5.2.1	Matching in Bipartite Graphs	76
5.2.2	Linear Problem Formulation of the Bipartite Matching . .	77
5.3	Combinatorial Subgraph Matching Approach	77
5.3.1	Quadratic Integer Program	78
5.3.2	Regularization Parameter	81
5.3.3	Hidden Parameters	82
5.3.4	Classification of the Approach	82
5.4	Semidefinite Program Formulation	82
5.4.1	Objective Function	83
5.4.2	Equality Constraints	84
5.4.3	Inequality Constraints	87
5.4.4	Post-Processing to obtain an Integer Solution	90
5.5	Illustrative Example	91
5.5.1	The Subgraph Matching Example	91
5.5.2	Linear Bipartite Matching Approach	92
5.5.3	SDP Subgraph Matching Approach	92
5.5.4	Problem Nature	94
5.6	Subgraph Matching Experiments	95
5.6.1	Creation of the Problem Instances	95
5.6.2	Problem Nature	97
5.6.3	Influence of the Regularization Parameter	98
5.6.4	Statistical Performance Investigation	103
5.7	Random Subgraph Matching Experiments	108
5.7.1	Creation of the Problem Instances	109
5.7.2	Problem Nature	110
5.7.3	Performance Investigation	110
5.8	Real World Examples	115
5.8.1	Creation of the Real World Examples	115
5.8.2	Results	116
5.9	Discussion	124
5.9.1	Computational Effort	125
5.9.2	Reducing the Computational Effort	127
5.9.3	Structural Perturbations	134
5.9.4	Bimodal Experiments	136
5.9.5	A New Bound for Subgraph Non-Isomorphism	138
6	Conclusion	141
6.1	Summary	141
6.1.1	Weighted Graph Matching	142
6.1.2	Subgraph Matching	144
6.2	Future Work	147

A Quadratic Assignment Supplements	151
A.1 The Dual of the relaxed homogeneous QAP	151
A.1.1 QAP Dual as linear program	152
A.1.2 Non unique solutions for the EVB	155
A.2 QAP-Bounds	157
B Subgraph Matching Supplements	159
B.1 Example Data	159
B.1.1 Illustrative Example Data	159
B.1.2 Graph Data for the Parameter Dependency Example . . .	160
B.2 Earth Movers Distance	161
Bibliography	164

Abstract

Automatic recognition of objects in images is a difficult and challenging task in computer vision which has been tackled in many different ways. Based on the powerful and widely used concept to represent objects and scenes as relational structures, the problem of graph matching, i.e. to find correspondences between two graphs is a part of the object recognition problem. Belonging to the field of combinatorial optimization graph matching is considered to be one of the most complex problems in computer vision: It is known to be NP-complete in the general case.

In this thesis, two novel approaches to the graph matching problem are proposed and investigated. They are based on recent progress in the mathematical literature on convex programming. Starting out from describing the desired matchings by suitable objective functions in terms of binary variables, relaxations of combinatorial constraints and an adequate adaption of the objective function lead to continuous convex optimization problems which can be solved without parameter tuning and in polynomial time. A subsequent post-processing step results in feasible, sub-optimal combinatorial solutions to the original decision problem.

In the first part of this thesis, the connection between specific graph-matching problems and the quadratic assignment problem is explored. In this case, the convex relaxation leads to a convex quadratic program, which is combined with a linear program for post-processing. Conditions under which the quadratic assignment representation is adequate from the computer vision point of view are investigated, along with attempts to relax these conditions by modifying the approach accordingly.

The second part of this work focuses directly on the matching of subgraphs – representing a model – to a considerably larger scene graph. A bipartite matching is extended with a quadratic regularization term to take into account relations within each set of vertices. Based on this convex relaxation, post-processing and the application to computer vision are investigated and discussed.

Numerical experiments reveal both the power and the limitations of the approach. For problems of sizes which occur in applications the approach is quite reasonable and often the combinatorial optimal solution is found. For larger instances the intrinsic combinatorial nature of the problem comes out and leads to sub-optimal solutions which, however, are still good.

Zusammenfassung

Die automatische Erkennung von Objekten in Bildern ist eine der größten Herausforderungen in der Bildverarbeitung. Werden die Objekte und Szenarien durch Graphen repräsentiert, ist ein Teilproblem der Objekterkennung die gewünschte Zuordnung der Knoten zweier Graphen zu finden. Dabei soll möglichst die Graphstruktur und evtl. zusätzlich vorhandene Information berücksichtigt werden. Die Bestimmung der besten Korrespondenzen der Graphknoten, auch Graph Matching genannt, ist für allgemeine Graphen ein NP-Hartes kombinatorisches Problem und gehört damit zu den schwierigsten aller Probleme in der Bildverarbeitung.

In dieser Arbeit führen wir zwei neue Ansätze ein, um Graph Matching und Subgraph Matching Probleme approximativ zu lösen. Dazu nutzen wir neuere Methoden und Erkenntnisse der konvexen Optimierung. Die Graph Matching Probleme können als 0/1-Integer Optimierungsproblem formuliert werden und lassen sich mit Hilfe einer geeigneten Relaxierung in ein kontinuierliches und konvexes Problem transformieren. Ein Vorteil konvexer Optimierungsprobleme liegt in der Tatsache, dass sie ohne zusätzliche Parameteroptimierung effizient mit Standardverfahren gelöst werden können. Ein Nachverarbeitungsschritt sorgt dafür, dass mit Hilfe der Approximierten Lösung eine für das Originalproblem passende und gute 0/1-Integer Lösung gefunden wird.

In dem ersten Teil dieser Arbeit untersuchen wir einen Ansatz, der das Problem des gewichteten Graph Matchings auf die Klasse von Quadratischen Assignment Problemen zurückführt. Die Relaxierung führt in diesem Fall zu einem konvexen quadratischen Optimierungsproblem. Um eine nahliegende kombinatorische Lösung zu erhalten, wird ein geeignetes lineares Optimierungsproblem nachgeschaltet. Wir untersuchen, in wie fern das Verfahren für Probleme der Bilderkennung geeignet ist und diskutieren mögliche Verbesserungen.

Im zweiten Teil dieser Arbeit konzentrieren wir uns auf das Problem des Subgraph Matchings wobei die Knoten eines kleineren Objektgraphens den Knoten eines größeren Szenen-Graphens zugeordnet werden sollen. Dazu erweitern wir ein sogenanntes Bipartites Matching um einen quadratischen Term, so dass neben der Ähnlichkeit der Knoten untereinander auch die zugrundeliegende Struktur der Graphen berücksichtigt wird. Anschließend untersuchen wir eine konvexe Relaxierung dieser Problemformulierung mit einem entsprechenden Nachverarbeitungsschritt und diskutieren verschiedene Einflüsse auf diesen Subgraph Matching Ansatz.

Unsere numerischen Experimente zeigen, dass unsere Verfahren meist sehr gute und oft optimale Lösungen finden. Bei größeren Problemen macht sich jedoch zunehmend die kombinatorische Natur der Probleme bemerkbar, trotzdem werden in der Regel suboptimale Lösungen sehr guter Qualität gefunden.

Acknowledgements

I am especially grateful to Prof. Christoph Schnörr who gave me the opportunity to work and to write my dissertation at the CVGPR group at the University of Mannheim. He introduced me into the wonderful field of computer vision and suggested the idea to solve computer vision problems by convex optimization techniques. I could always count on his support and advice. In enthusiastically and inspiring discussions he helped to find solutions for seemingly desperate problems. He also gave valuable comments on the first versions of this thesis. I would like to thank Prof. Joachim Denzler for serving as an external referee of this thesis.

I am grateful to Prof. Waldemar Rohde for the opportunity to teach several programming courses at the Fachhochschule Südwestfalen in NRW.

I am also grateful to Joachim Weickert who introduced me into several aspects of computer vision like pde-based approaches in computer vision. Furthermore, I thank Joachim Hornegger who introduced me into the interesting field of 3D-reconstruction.

I am very thankful to Jens Keuchel and Christian Gosch who read preliminary versions of this work and helped with their comments to improve the readability of this thesis.

I would like to acknowledge the work of Stefan Roth who investigated a deterministic annealing graph matching approach in his diploma thesis.

The working atmosphere at the CVGPR group has been very pleasant and I would like to thank all members of the group for the inspiring and enjoyable environment. I specially value the friendship of many of my colleagues: Daniel Cremers, Christian Gosch, Jens Keuchel, Timo Kohlberger, Paul Ruhnau, Thomas Schüle, Annette Stahl, Stefan Weber and Jing Yuan. They made several trips and events unforgettable. Special thanks to all the members of the group that made the system administration easier by being patient and uncomplaining when a problem occurred. In addition, I would like to thank Hendrik Lemelson to handle the hardware related side of the system administration at the group. Furthermore I would like to thank Mrs. Schieker for her help to handle the large amount of administration effort.

I am extremely grateful to my parents Armin and Ursula Schellewald, my brothers Andreas and Martin Schellewald and my grandparents Anneliese and Josef Jelovsek and Hildegard Schellewald who gave me both the freedom and support that I needed to become the person I am.

I want to thank all my friends who forced me to participate in life outside science. Especially I want to thank Oliver Scholle, Maike Scholle and Bettina Russ for their long time friendship.

Last but not least I want to thank my girlfriend Annette Stahl who encouraged and supported me during the last year.

Chapter 1

Introduction

In this chapter we motivate the use of graph matching algorithms in computer vision. After sketching the basic idea to apply convex optimization techniques for finding approximate solutions to the combinatorial hard problem of graph matching we summarize the contribution of this work. Then an outline of this thesis is given followed by the enumeration of related approaches used for graph matching.

1.1 Motivation

The automatic visual recognition of objects is regarded as one of the most difficult problems in computer vision and is at the same time one of the most fascinating challenges in computer science. In this thesis object recognition is approached in a way that leads to the problem to find good matchings between relational structures. Several methods for the representation of object views have been proposed, e.g. pixel based or shape based representation. This thesis follows the wide used approach to represent object views and scene views as relational structures, namely as graphs.

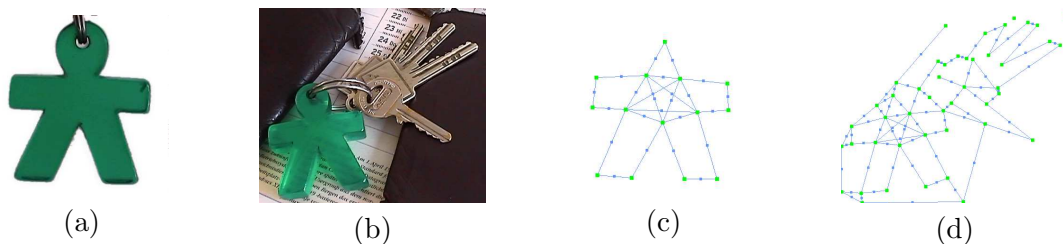


Figure 1.1: View based object recognition. The object view (a) and the scene view (b) are represented as view based graphs in (c) and (d), respectively. This turns the object recognition task into a sub-graph matching problem.

A common way to create view-based graphs is to use a set of image features together with pairwise relations between them. The image features could be, for example, points, line segments, or curve segments which are found by an appropriate feature detector in a low-level pre-processing phase.

An example for an object recognition problem is shown in figure 1.1. The first two pictures, a) and b) shows a view of an object and a scene view containing the object, respectively. The aim of the recognition task is to detect the given object within the scene. The task of recognizing an object within a scene can be traced back to the problem of finding a good matching between two graphs. In this thesis we restrict ourself to the problem to find good matchings between relational structures. We assume that the object view and scene view are already represented by graphs. The pictures c) and d) of figure 1.1 are examples for the representation of the views as view based graphs. As the representation of relational structures is also used in other fields like biometric identification, document processing, networks and others, the investigations within this thesis maybe useful in these fields too.

1.2 Overview

This thesis contributes to the field of object recognition in computer vision. Using a graph representation of object and scene views, the object recognition problem turns into the problem of finding correspondences between graphs. Therefore this work is concerned with the problem to find good matchings between pairs of graphs. Such problems can be stated appropriately by combinatorial optimization problems. The primary motivation for our work is the design of algorithms for which the performance does not critically depend on the choice of tuning parameters, since the automatic choice of suitable parameter values is rather difficult in the context of computer vision systems. To obtain such parameter free algorithms, convex programming approaches are an appropriate choice. Before discussing this idea in the context of graph matching, some basics related to graph matching are briefly introduced. For a more detailed introduction we refer to chapter 2.

1.2.1 Graph Matching

A graph consists of nodes and edges where the edges represent pairwise relations between the nodes. The expression *graph matching* in this thesis refers to the problem to find a mapping between two sets of nodes which belong to two different graphs. The matchings we are interested in are called *maximal bipartite matchings*. These represent mappings where every node of the possibly smaller graph is linked to exactly one node in the other graph, with no two nodes being linked to the same node. This requirement is often referred to as *matching constraint*.

In figure 1.2 we show an example of a bipartite matching between two graphs. The bipartite matching is represented by the green line segments. Each line connects two nodes from the different graphs which are mapped to each other. Our graph matching problems can be stated as discrete optimization problems which model the matching constraint appropriately.

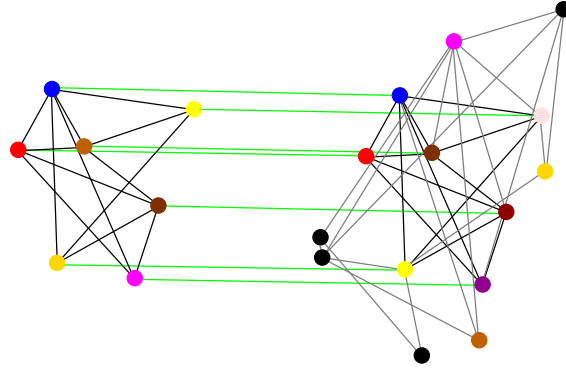


Figure 1.2: The bipartite matching (green line segments) represents the mapping between the two sets of nodes belonging to the two different graphs.

1.2.2 Exact and Inexact Graph Matching

Graph matching problems can be divided into two problem classes: *Exact* and *inexact* graph matching problems. Exact graph matching refers to (sub)graph isomorphism problems while inexact graph matching refers to all problems where an exact isomorphism cannot be expected to occur. To be more precise, *exact graph matching* is the problem to find an exact one to one mapping between the nodes of two graphs such that each edge of one graph is mapped to an identical edge of the other graph. If such a mapping exists the graphs are called to be isomorphic. However, due to changed views or noise it is unlikely that graphs extracted e.g. from vision systems are isomorphic to model graphs. Therefore for real world problems the class of inexact graph matching problems is much more relevant. The problem of finding an inexact graph matching is usually formulated as an optimization problem with an objective function which measures the quality of a match.

1.2.3 What this thesis is about

Unfortunately general graph matching problems are NP-hard combinatorial optimization problems that are intractable for graphs even of medium size (e.g. more than 20 vertices) on today's computers. However, due to the applicability of graph matching in computer vision and other fields there is a high interest in good approximation algorithms. These algorithms should be able to compute good suboptimal solutions – in contrast to optimal solutions – and should have a polynomial time complexity. In this work, we focus on *convex optimization techniques* in order to find good approximations for graph matching problems. In the context of combinatorial optimization problems a *convex relaxation* can be obtained by relaxing the integer constraints along with an appropriate reformulation of the original objective function into a convex approximation of the problem. Such a convex optimization problem involves no additional tuning parameters and can be solved in polynomial time with standard methods like *interior point* algorithms. The obtained solution of the relaxed problem repre-

sents an approximate solution to the original problem. With a post-processing step a feasible and hopefully good integer solution of the original problem can be calculated. The global optimization of the relaxed convex optimization problem makes it likely that bad local minima of the original integer problem are avoided.

1.2.4 What this thesis is not about

Although this is an important issue, we would like to point out that this thesis is not concerned with the image preprocessing question of how a vision system can extract scene and object graphs in a reliable way. We assume that the object graphs and the scene graphs for which we want to find a good matching are already represented as graphs. The implication of this assumption is that the used graphs in this thesis are either created randomly, by hand or just by a triangulation of features extracted in a picture by a simple feature detector. Performance investigations of the graph matching approaches are usually made with randomly created graphs while shown examples are often manually created.

1.3 Convex Optimization

Many problems in computer vision can be stated as optimization problems, but unfortunately, often the thorough mathematical models results in intractable NP-hard problems. This leads to the idea – which we pursue in this thesis – to approximate such intractable problems by convex and continuous problem formulations for which the solution can be efficiently computed. The fundamental properties which makes convex and continuous approximation approaches favourable are summarized here:

- A convex optimization problem has no local minima which is not the global minimum and therefore one can not be trapped in a bad local minimum.
- Under mild conditions a convex and continuous optimization problem can be efficiently solved to optimality.
- No tuning parameters are introduced which can critically influence the quality of the solution.
- Often a non-convex combinatorial optimization problem can be approximated by an appropriate convexified optimization problem.

These properties are utilized in this thesis to propose approaches for graph and subgraph matching problems which are theoretically clear and free of tuning parameters.

Several excellent books concerned with convex optimization techniques have been published recently e.g. by Boyd and Vandenberghe [20], Bertsekas [15] and Ben-Tal and Nemirovski [12].

1.3.1 Semidefinite Programming

Semidefinite programming (SDP) represents a special type of convex optimization problems and can be seen as unification of linear and (convex) quadratically constrained quadratic programming as it includes both as special cases [140]. The origin of semidefinite programming goes back to Bellman and Fan [11] in 1963 who discussed theoretical properties of semidefinite programs. It is an ongoing active area of research and in recent years a very successful and wide applicable convex relaxation technique called *lifting procedure* for binary quadratic optimization problems has emerged which leads to semidefinite programs. Poljak described this technique in [115] which aims to approximate the exact solutions for binary quadratic optimization problems. This scheme was introduced by Lovász and Schrijver [96] and Lemaréchal and Oustry [95] showed that this technique has its roots in Lagrangian duality. It has been applied to a wide range of combinatorial optimization problems: max-cut and max-2sat [54], the quadratic knapsack problem [68], segmentation and image restoration problems [83]. More applications can be found e.g. in [146].

As the combinatorial optimization problems in this thesis are defined as quadratic 0/1-integer problems the SDP relaxation recipe [115] can be applied to the combinatorial graph and subgraph matching problems we are concerned with in this thesis.

Note that in principle semidefinite programs can be solved in polynomial time by the ellipsoid algorithm [58] but interior point algorithms turned out to be the faster alternative [2, 109]. Due to the huge research interest in semidefinite programming today several reliable SDP solvers are available (e.g [13, 92, 18]). An independent benchmarking for many different SDP solvers can be found in [104].

1.4 Contribution

This work has the aim to investigate if the application of convex relaxation methods is a reasonable approach in object recognition to cope with the generally NP-hard problem of inexact graph matching. It has two main contributions. The first is the application of a particular convex relaxation developed for the quadratic assignment problem in order to find good correspondences between two weighted graphs. The second contribution lies in the formulation of a combinatorial subgraph matching approach which allows the application of the convex semidefinite relaxation technique. These two approaches are outlined below.

1.4.1 QAP Convex Relaxation used for Graph Matching

The quadratic assignment problem (QAP) is a well known combinatorial optimization problem. It has attracted our interest as some graph matching problems can be formulated as QAP. Recent and ongoing research in that field has led to relaxations which allow the calculation of good lower bounds together with approximate solutions for QAPs. Following the work of Anstreicher, Brix-

ius and Wolkowicz [5, 21] we have explored their convex relaxation approach for QAPs in view of its capability to find the desired correspondences between pairs of weighted graphs. The investigation of this approach has been published in the DAGM2001 conference proceedings [126].

1.4.2 Subgraph Matching by Semidefinite Programming

As the previous convex relaxation approach fails to be applicable in its original form to graph matching problems with different sized graphs, we searched for an applicable subgraph matching approach. We developed a combinatorial subgraph matching approach which incorporates the relational constraints of both graphs and can be approximately solved by convex optimization methods. Starting from a linear programming formulation for the computation of optimal bipartite matchings, we extend the objective function by a quadratic term in order to take into account the relational constraints given by both graphs. The resulting combinatorial optimization problem is approximately solved by a (convex) semidefinite program. Preliminary results have been published in the IWCIA2003 conference proceedings [127].

1.5 Outline

In the following we outline the structure of the remainder of this work.

- This chapter 1 ends with a summary of related work in the field of graph matching.
- In chapter 2 we introduce the basic graph concepts that are needed to define the graph matching problems we are concerned with in this thesis. We focus on maximal bipartite matchings which build the basis for the subgraph matching approach in chapter 5 and illustrate how we generally represent matchings in this work. After discussing the complexity of our graph matching problems we provide the used notation and introduce some mathematical definitions. Then some basic convexity concepts are summarized.
- In chapter 3 we discuss relaxation techniques which lie behind the convex relaxations which are used to approximate the intractable combinatorial graph matching problems. Furthermore we discuss optimality conditions for convex optimization problems and outline the convex optimization problems that appear within this thesis. Then we discuss how linear programming can be exploited to compute feasible 0/1-integer solutions from the approximated solutions.
- In chapter 4 we study the application of a particular non-trivial convex relaxation approach to the problem of weighted graph matching. We discuss several relaxations to this problem and explain in detail the convex

relaxation of Anstreicher and Brixius who proposed this relaxation for the quadratic assignment problem. To compute a feasible combinatorial solution we propose an improved post processing step which is based on a linearization of the original problem. A small graph matching problem is used to illustrate the resulting convex relaxation. Several statistical experiments including ground truth experiments show the performance of this approach. A real world experiment demonstrates the applicability to computer vision related problems. Then we outline how this approach can be adapted to be able to cope with weighted subgraph matching problems as well.

- In chapter 5 we propose a new combinatorial subgraph matching approach which is based on a quadratic extension of the linear bipartite graph matching approach to incorporate the structures of both the object and the scene graph. The 0/1-quadratic integer problem formulation allows to obtain a convex semidefinite relaxation for this combinatorial problem which is explained in detail. Subsequently we investigate thoroughly the capability of this subgraph matching approach and present several real world problems in computer vision. Then we discuss possible computational improvements and investigate the approach in some interesting circumstances. Furthermore we propose to utilize the lower bound computed by our subgraph matching approach as an indicator for subgraph non-isomorphism.
- In chapter 6 we conclude with a summary of our work and provide an outline of several promising future research directions.

1.6 Related Work

Graph matching has a history of more than thirty years in pattern recognition [30]. One of the first publications that suggests graphs for image representation was published in 1971 by Barrow and Popplestone [9]. Recently graph matching has found increasing interest as the computational power increases and it is now possible to tackle graph based algorithms for graphs with sizes that are reasonable for computer vision problems. In section 1.6.1 we outline several basic approaches that have been applied directly to tackle different kinds of graph matching problems. In section 1.6.2 we sketch some more general concepts which have been successfully used to state graph matching problems.

1.6.1 Basic Graph Matching Approaches

In the following we summarize basic approaches which have been used for solving graph matching problems. For these approaches the properties of interest are summarized in table 1.1.

	global optimum	continuous/discrete approach	probabilistic approach
Tree search/Branch and Bound	yes	discrete	no
Genetic based Search	no	discrete	no
Discrete Relaxation	no	discrete	no
Continuous Relaxation labeling	no	continuous	(yes)
Simulated Annealing	yes	discrete	yes
Deterministic Annealing	no	discrete	yes
Expectation Maximization	no	continuous	yes
Spectral approaches	no	continuous	no

Table 1.1: Classification and Properties of various graph matching approaches

Tree Search/Branch and Bound

Many algorithms for graph matching are based on a *tree search* with backtracking. In these approaches one iteratively tries to enlarge a partial match. If an iteration is reached where the partial match cannot be further expanded, conform to the matching constraints, the algorithm *backtracks*. That is, it turns back to a partial match where an alternative choice for an extension can be made. If all possible matchings have been tried, or a graph matching is found, the algorithm stops. One of the first and most popular tree search algorithm was proposed by Ullmann [138] in 1976 and addresses exact graph matching problems. It uses a so called *refinement procedure* to eliminate branches in the search tree that can not lead to subgraph isomorphism.

A well known approach in tree search is the *branch and bound* approach. By calculating lower bounds one tries to cut off subtrees that will not be searched. To cut off search branches as early as possible one needs tight lower bounds, which is one reason that researchers are interested in good lower bounds. The branch and bound approach is guaranteed to find the optimal solution, but the complexity in the worst case is as high as that of exhaustive search.

Evolutionary Strategies

Genetic algorithms are inspired by the natural evolution process and are a kind of stochastic search methods. Usually they define some genetic operators, like reproduction, crossover and mutation, to create the next generation of proposed

solutions. The selection process is based on a fitness value which measures how good a proposed solution is. Individual solutions with a higher fitness value have a higher probability to survive. Stop criteria are often a limited generation number or the time no improvement to the best solution is achieved. The first book about genetic algorithms was published in 1975 by Holland [73]. In 1997 Cross, Wilson and Hancock [34] used genetic search for inexact graph matching. A method called *extremal optimization* belongs also into this category. The extremal optimization approach selects the weakest solution in a solution pool for adaptive changes to improve its fitness. Beside the application to graph partitioning and the traveling salesman problem (Boettcher and Percus [16]) it has been applied to find point correspondences by Meshoul and Batouche [100].

Consistency Algorithms

The *consistency algorithms* for graph matching can be divided into discrete and continuous approaches. The *continuous relaxation labeling* is an enhancement of the *discrete relaxation* which allows a probabilistic interpretation for the assignment of the nodes in a graph. In the following we explain both approaches.

Discrete Relaxation

Discrete relaxation algorithms or more accurately *discrete consistency algorithms* were first introduced by Waltz [141] in 1975 for the interpretation of scenes with objects that cast shadows. In the original algorithm possible discrete labels of nodes are iteratively removed such that the nodes contain only labels which are consistent with some neighbored nodes constraints. Haralick and Shapiro defined an operator that reduced the size of the tree searched for a consistent labeling [62]. In the discrete relaxation approach of Kim and Kak [85] in 1991 inconsistent models are rejected by using bipartite matching to determine the compatibility of a scene with a possible model. More recently, Wilson and Hancock [143] describe a Bayesian framework for performing relational graph matching by discrete relaxation.

Continuous Relaxation labeling

The term *relaxation labeling* goes back to a publication by Rosenfeld, Hummel and Zucker [121] in 1976 with the title 'Scene Labeling by Relaxation Operations'. They enhanced the *Discrete relaxation* of Waltz [141] by introducing a so-called *stochastic labeling* which requires that the sum of the label weights for each component is one. This allows a probabilistic interpretation of the weights. *Relaxation labeling* or *probabilistic relaxation* iteratively refines the (continuous) probability of class labels that can be assigned. The refinement takes the label-probability of neighbored components into account. To ensure a discrete labeling as solution a maximum selection (winner-take-all) can be used. Faugeras and Berthod [42] defined a minimization problem by using transition probabilities. Kittler and Hancock [87] justified the relaxation labeling approach 1989 theoretically by casting the process into a Bayesian framework. Often relaxation labeling is used for classification tasks where labels can occur more than once. In the case of graph matching this means there is no guarantee

that each node of the first graph is assigned to a different node of the second graph. Christmas [28] used *probabilistic relaxation* for structural matching like road matching and the matching of edge segments in a stereo pair of images.

Annealing

Annealing approaches try to minimize problem specific objective functions which, in the context of physics, often represent the total energy of a system. Annealing approaches can be divided into *simulated annealing* and *deterministic annealing* approaches depending on the method used to create/accept a new valid system configuration.

Simulated Annealing

Simulated annealing algorithms have their origin in physics, namely in statistical mechanics. The name is deduced from a slow cooling process for crystal structures (solids) in order to reach a state with minimal energy. The first simulated annealing algorithm goes back to 1983 when Kirkpatrick, Gelatt and Vecchi [86] published the well known paper with the title “Optimization by simulated annealing”. Geman and Geman [51] determined in 1984 an annealing schedule that is sufficient for convergence. A simulated annealing algorithm starts with a valid system configuration (state) at high temperature. A new state of the system is generated by a random perturbation of the system which magnitude is dependent from a temperature parameter. If the energy of the new state is lower than the previous state-energy, the new system state is accepted. If the energy is greater, the new state is only accepted with a probability related with the energy-difference. This allows the system to get out of local minima. In an annealing schedule the temperature is lowered slowly until the system converges. If the annealing schedule decreases the temperature logarithmically, simulated annealing guarantees an optimal solution [57]. But the need of a logarithmic decrease of the temperature in the annealing schedule leads to an impractical running time for these algorithms. Herault and Horaud [71] have used the simulated annealing for performing structural matching tasks.

Deterministic Annealing

Deterministic annealing approaches are using annealing schedules like simulated annealing, but are – in contrast to simulated annealing – using deterministic rules for the acceptance of a new state. Deterministic annealing does not guarantee a global optimum and it is still an open question, if deterministic annealing can be set up to converge globally. But it is able to avoid bad local minima. References to deterministic annealing approaches in different fields can be found in a work from Hofmann and Buhmann [72] who have used deterministic annealing for clustering. A deterministic annealing approach for graph matching has been proposed by Gold and Rangarajan [55]. Their algorithm is called *graduated assignment* algorithm. It incorporates a method discovered by Sinkhorn [132] which is used to satisfy the two-way assignment constraint that one graph node is matched to at most one node in the other graph and vice versa. Independently Ishii and Sato [77] developed a very similar *doubly con-*

straint network algorithm. Rangarajan [119] has introduced an approach called *self annealing* which is closely related to deterministic annealing. The algorithm has the goal to eliminate the temperature schedule by finding a schedule itself. The self annealing approach provides also a link to relaxation labeling, e.g. the original relaxation labeling algorithm by Rosenfeld et al. [121] can be seen as an approximation of the self annealing algorithm.

Expectation Maximization

The Expectation Maximization (EM) algorithm is an iterative statistical algorithm for computing maximum likelihood parameter estimates from some incomplete data. The EM algorithm repeats two steps, the *Estimation*-step and the *Maximization*-step, until it converges. One early paper on *Maximum likelihood estimation from incomplete data* is published 1958 by Hartley [65]. The formalization and a convergence proof of the EM algorithm can be found in the paper of Dempster, Laird and Rubin [36] in 1977. Usually only very few parameters have to be tuned and some constraints like positivity are automatically satisfied. But there is no guarantee that the EM algorithm escapes bad local minima. In 1998 Cross and Hancock [33] used the EM framework to recover the transformational geometry and node correspondences of relational graphs. They iteratively used the maximized likelihood of the transformation parameters to improve the accuracy of the point correspondences. The improved point correspondences are then used to improve the transformation parameters. Luo and Hancock [97] also used the EM algorithm to recover correspondence matches based on singular value decomposition for efficiently finding the direction in the *Maximization*-step.

Spectral Approaches

Spectral approaches are using eigendecompositions of the adjacency matrices of the graphs to compute a good matching. Umeyama [139] showed in 1988 that a desired matching could be found efficiently if the graphs are sufficiently close to each other.

Concave Minimization

In 2003 Maciel and Costeira [98] proposed a new method based on *concave optimization* to solve the point correspondence problem. They reformulated the integer optimization problem as a concave objective function and relaxed the search domain into its convex hull. The concave optimization problem can be solved efficiently and the special structure assures that it is equivalent to the original problem.

1.6.2 More General Graph Matching Approaches

In this section we summarize three wider concepts which have been used in the context of graph matching problems. Within these concept classes many of the

basic methods mentioned above have been applied to solve the problems specified by these classes. The first class which is directly related to graph matching consists of *error-correcting* graph matching approaches which minimize some graph *edit costs* to find node correspondences between two graphs. The second class is concerned with the problem of finding the largest fully connected subgraph (i.e. the *maximum clique*) of an *association graph*. The maximum clique problem is equivalent to the problem of finding the maximum common subgraph of two graphs. The *quadratic assignment problem* represents a third problem class which can be used for graph matching. We use this problem formulation in chapter 4 of this thesis for finding correspondences between two weighted graphs.

Error-Correcting Graph Matching

Error-correcting graph matching also known as *error-tolerant* graph matching calculates an assignment between nodes of two graphs based on the minimization of graph *edit costs*. One tries to find a sequence of graph edit operations with a minimal cost which transforms one of the graphs into the other. Commonly introduced graph edit operations are, for example, deletion, insertion, and substitution of nodes and edges. Each graph edit operation has a cost assigned which is application dependent. Generally the edit cost is chosen smaller if a certain distortion of the graph is more likely to occur. The minimal graph edit cost defines the so called *edit distance* between two graphs. The idea to define the edit distance for graph matching goes back to Sanfeliu and Fu [125] in 1983. The edit distance was used for string matching before. Algorithms for error correcting graph matching are based on different methods like tree search [102], genetic algorithms [142] and others (see [23]). The subgraph matching approach we propose in chapter 5 belongs into this class of graph matching approaches as it can be interpreted as a minimization of a graph edit cost.

Maximum Clique Problem

The problem of finding the maximum common subgraph of two graphs can be reformulated into the so called *maximum clique* problem. This problem consists in seeking the largest clique i.e. the largest fully connected subgraph in an appropriately defined *association graph*. Each node in an association graph represents a possible assignment between one node of the first graph and one node of the second graph. If two such matchings (two nodes in the association graph) are compatible in terms of the matching constraints, they are connected by an edge. The largest fully connected subgraph of this graph corresponds to the maximum common subgraph. This idea goes back to Ambler et al. [4] and has been applied to many different computer vision problems (e.g. shape matching [114], stereo correspondence [75], motion analysis[116]). A survey to the maximum clique problem can be found in Bomze et al. [17]. One important technique to solve the maximum clique problem, published by Pardalos and Phillips [111] in 1990, is due to the Motzkin-Straus [105] theorem which allows to formulate the maximum clique problem as continuous quadratic op-

timization problem. Approaches that have been applied to solve the maximum clique problem are discussed in [17] in more detail (e.g. genetic algorithms [26], simulated annealing [78]).

Quadratic Assignment Problem

In chapter 4 of this thesis we investigate a graph matching approach that is related to the *quadratic assignment problem* (QAP), which is a well known combinatorial optimization problem. The QAP can be illustrated by considering the problem to assign a number of factories to a number of places. This results in total costs which depend on the factory building costs in connection with costs related to exchanged material. The aim is to minimize the total cost. The optimization of quadratic assignment problems is a huge field of its own and has been investigated by many researchers. It has been introduced by Koopmans and Beckmann [90] in 1957. A survey of the quadratic assignment problem can be found in Burkard et al. [24]. Sahni and Gonzalez [124] showed in 1976 that this problem is NP-complete. The approaches to solve quadratic assignment problems can be divided in non-relaxation approaches and relaxation approaches. The latter approaches provide a method to compute lower bounds to the QAPs.

- **Non-Relaxation Approaches:** In 1988 Umeyama [139] developed an algorithm which used the eigendecompositions of the adjacency matrices of the graphs to solve a graph matching problem. It is closely related to QAPs as the weighted graph matching problem formulated in [139] can be reformulated into a QAP. In 1996 Gold and Rangarajan [55] used the *graduated assignment* algorithm, which is a deterministic annealing approach, for graph matching problems. Their objective function is stated similar to the QAP. Independently Ishii and Sato [77] published in 2001 a very similar so called *doubly constrained network* approach.
- **Relaxation Approaches:** In 1987 the authors Finke, Burkard, Rendl [44] proposed an eigenvalue bound for quadratic assignment problems which turns out to represent a weak lower bound for QAPs. The advantage of this approach is that the corresponding matrix for which the bound is achieved can be easily calculated. An improved bound can be computed by the projected eigenvalue bound suggested by Hadley, Rendl, Wolkowicz [59] in 1992. But only for special cases an explicit matrix for which the projected eigenvalue bound is achieved can be calculated. Anstreicher and Brixius [21] overcame this drawback in 1999 by developing the convex quadratic programming bound which additionally further improved the projected eigenvalue bound.

Chapter 2

Preliminaries

In this chapter we first introduce some relevant terms related to graph matching. To recall how hard the problem of generic graph matching is to solve, the complexity concepts for finite sets are recapitulated. Then some mathematical preliminaries are summarized. A reader familiar with the summarized concepts can skip the known parts.

2.1 Basic Graph Theory

In this section we introduce the basic concepts of graphs that are used throughout this thesis. An *undirected, weighted graph* $G = (V, E, w)$ consists of n *vertices (nodes)* $V = \{1, \dots, n\}$, *edges* $E \subset V \times V$ and a weight function $w : E \rightarrow \mathbb{R}_0^+$ which defines the *edge weights*. The number of vertices n is also denoted by $|V|$. A graph that contains no edges starting and ending at the same node (*self-loops*) is called a *simple* graph. Two vertices i and j are *adjacent*, or *neighboring*, if (i, j) is an edge of the graph. The vertices i and j are then called to be *incident* with the edge (i, j) . The structure and the weights of the undirected graph G can be represented by a symmetric *adjacency matrix* $A_G = A_G^T$. The elements of A_G are $(A_G)_{ij} = w_{ij}$, for $i, j = 1, \dots, n$. The edge weight is $w_{ij} = 0$ if the edge (i, j) is not present. For non-weighted graphs the weight is $w_{ij} = 1$ if the edge (i, j) exists and the appendant adjacency matrix is a 0/1-matrix. A small sample graph and the corresponding adjacency matrix are shown in figure 2.1.

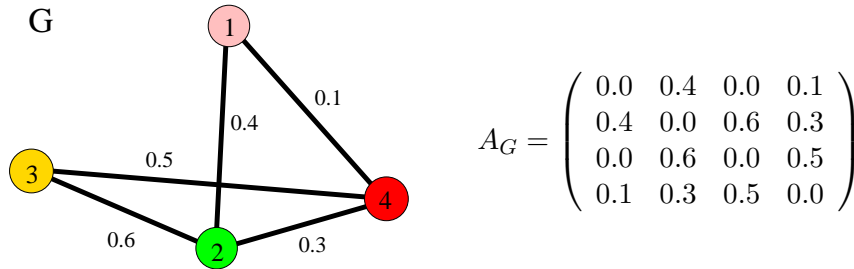


Figure 2.1: Example for a small weighted graph and its adjacency matrix

If the vertices or edges of the graph contain additional information, which e.g. could be represented as vector, the graph is called an *attributed graph*. For our applications, we assume that edge weights and node weights are scalar values. Furthermore, we assume that there is a scalar distance function so that vertices or edges of different graphs can be compared.

2.2 Graph Matching

As we are interested in object recognition we are mainly concerned with computing good mappings between sets of nodes which belong to two high-level graphs representing the object and the scene. High-level graphs refer to graphs with a manageable size which are used to describe object prototypes or scenes based on segmentation or features of an image. In contrast to that segmentation approaches often work on “low-level” graphs with thousands of nodes, where each node corresponds to a single pixel. The desired mapping of an object graph to a scene graph is constraint by several requirements. For example should every node of the probably smaller object graph be mapped to a different node of the larger scene graph. In the following we define how we use the term graph matching in this work and how the matchings and related solutions are represented.

2.2.1 Bipartite Matching

In the literature *matching* is the problem to find an one to one correspondence between vertices so that each vertex is incident with at most one edge. In other words, no vertex is linked to more than one other vertex (see e.g. [53]). For a natural description of correspondences between two sets of nodes a *bipartite matching* can be used. Bipartite matching combines the idea of matching and the concept of a so called *bipartite graph*. A bipartite graph is an undirected graph where the vertices can be divided into two sets such that no edge connects vertices in the same set. In our case the two sets are predetermined by the vertices of the object and the scene graph, respectively. Assuming that the nodes of the smaller graph should be mapped to nodes of the other graph a so called *maximal bipartite matching* is a matching where each vertex of the smaller set is matched to exact one different vertex in the second set.

To simplify matters, we will often use the term matching or bipartite matching instead of maximal bipartite matching in this thesis.

2.2.2 Representation of Matchings

We illustrate how matchings are represented in this work. Figure 2.2 shows a maximal bipartite matching which represents the correspondences between two small sets of vertices. The corresponding bipartite graph consists of the colored edges and incident nodes. In this example the following mapping of nodes is depicted: $1 \mapsto 2$, $2 \mapsto 1$, $3 \mapsto 4$.

The first set with $m = 3$ vertices can be assumed to belong to the object graph. The second set with $n = 5$ vertices corresponds to the scene graph. In general

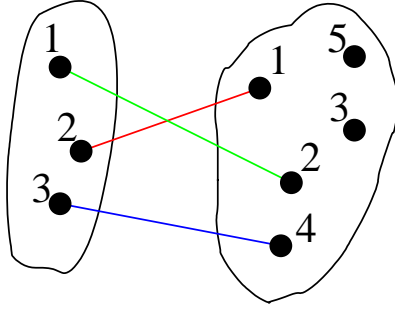


Figure 2.2: A bipartite matching between two vertex sets. The smaller set can be assumed to represent the nodes of an object graph which are mapped against the nodes of a scene graph in the second set. In this thesis we often show the computed mappings of corresponding nodes by a bipartite graph which is here depicted by the colored edges and incident nodes.

we assume that the object graph is not larger than the scene graph ($m \leq n$). In this thesis we use binary matrices and vectors to represent the matchings between the nodes of the two graphs in a mathematical way. These representations are sketched in the following.

Matching/Permutation Matrices

The one to one correspondences between two sets of vertices with m and n vertices can be represented by a $n \times m$ matching matrix X with binary elements. The element X_{ij} is set to $X_{ij} = 1$ if the node i of the first set is matched to the node j in the second set. Otherwise X_{ij} is zero ($X_{ij} = 0$). Assuming that $m \leq n$, all column sums of X are one: $\sum_{i=1}^n X_{ij} = 1$ for $j = 1, \dots, m$. For matching matrices the row sum is not fixed and can be either 1 or 0. With $X \in \mathcal{M}^{n \times m}$ we denote the set of all $n \times m$ matching matrices. The appropriate matching matrix for the matching displayed in figure 2.2 is depicted here:

$$X = \begin{pmatrix} 0 & \textcolor{red}{1} & 0 \\ \textcolor{green}{1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \textcolor{blue}{1} \\ 0 & 0 & 0 \end{pmatrix}$$

If $m = n$ the matching matrix turns out to become a permutation matrix where besides the column sums all row sums are one, too. Hence, for permutation matrices denoted by Π we have the two conditions: $\sum_{i=1}^n X_{ij} = 1$ for $j = 1, \dots, n$ and $\sum_{j=1}^n X_{ij} = 1$ for $i = 1, \dots, n$.

Indicator Vector

Beside the representation as matching or permutation matrix we use an indicator vector $x \in \{0, 1\}^{nm}$ to represent the matchings. The vector form is closely

related to the matrix representation. The indicator vector x is built by appending the columns of the matching matrix X to each other. This alteration is formally done by the operator $\text{vec}[\cdot]$ (see 2.4.2). The elements of the indicator vector $x \in \{1, 0\}^{nm}$ are ordered as follows:

$$x = \text{vec}[X] = (X_{11}, \dots, X_{n1}, X_{12}, \dots, X_{n2}, \dots, X_{1m}, \dots, X_{nm})^\top.$$

As before, an existing edge between the vertex i of the first set and the vertex j of the second set is represented by $x_{ij} = 1$, otherwise it is $x_{ij} = 0$. With these definitions the indicator vector

$$x = (0, \textcolor{green}{1}, 0, 0, 0, \textcolor{red}{1}, 0, 0, 0, 0, 0, 0, 0, 0, \textcolor{blue}{1}, 0)^\top$$

represents the bipartite matching shown in figure 2.2. When presenting results we sometimes plot the elements of the indicator vector in a 2D plot like it is shown in figure 2.3. In these drawings the elements of the indicator vector

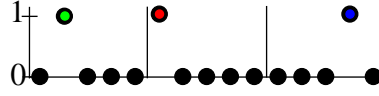


Figure 2.3: The 0/1-indicator vector represents the matching which is shown in figure 2.2.

are separated by vertical lines into m partitions of length n . According to the definition of x the j -th partition ($j = 1, \dots, m$) represents all possible matchings from the j -th vertex of the (smaller) first set to all n vertices of the (larger) second set. For a matching one requires in each partition exact one element to be equal to 1 while the others are 0. The approximation of an indicator vector can be represented in the same way, but is usually allowed to have element values in the range $[0, 1]$.

2.2.3 Classification of Graph Matching Problems

Graph matching problems can be divided into two problem fields. The first class consists of *exact graph matching problems* which require that the structure or the substructure of two compared graphs are identical. For pattern recognition the problems of the second class – *inexact graph matching problems* – are much more important. In such a case one tries to find a matching between two graphs even if the structure of the graphs differ to some extent. We describe the two classes below.

Exact Graph Matching

An example for an exact graph matching problem is the problem to find an one to one mapping $f : V_G \rightarrow V_H$ of vertices of the two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ such that each edge (u, v) of G is mapped to a corresponding edge $(f(u), f(v))$ in H . G is called to be *isomorph* to H if such a mapping

exists. Note that we have presumed that both graphs have the same number of vertices $|V_G| = |V_H|$.

A mapping is called *homomorphism* if all edges E_G in G can be mapped to edges E_H in H but the converse is not true. This means that H can have some extra edges.

The problem of finding the maximum common subgraph is also seen as an exact graph matching problem. Here the number of vertices in the two graphs can be different $|V_G| \neq |V_H|$ and one wants to find the largest subgraph of the first graph which is isomorphic to a subgraph of the second one.

Although it is not known if the general graph isomorphism problem can be solved in polynomial time [50], yet for some special graph types like planar graphs efficient algorithms are known [74, 40, 101].

Exact graph matching in this form is only of limited usability in computer vision. As the object and scene graphs obtained by a vision system are – due to noise or occlusion – likely different to some extent one cannot expect the graphs or subgraphs to be isomorphic in computer vision related problems. This gives rise to utilize *inexact graph matching* approaches.

Inexact Graph Matching

The graph matching problems we are concerned with in chapter 4 and chapter 5 both belong to the class of inexact graph matching problems. This class includes all graph matching problems which cannot be seen as exact graph matching problems. Therefore this category includes most of the graph matching problem instances which arise in computer vision. The quality of an inexact matching is usually defined by an appropriate objective function and the aim of the graph matching algorithms is to find the matching which minimizes (or maximizes) the value of the objective function. The graph edit distance in error correcting graph matching approaches is an example for such an objective function which has to be minimized to find the best matching.

2.3 Complexity

The complexity theory is concerned with the study of the intrinsic complexity of computational problems. The graph matching problems considered in this work are easy to formulate but belong to a problem class which is extremely hard to solve. Until today, there is no polynomial time algorithm known which can guarantee to find the optimal solution for this kind of problems.

The concept of polynomial time algorithms was introduced by Edmonds [39] and Cobham [29] in 1964 and the complexity of problems can be classified in classes like P, NP and NP-complete since the pioneering publications of Cook [31] and Karp [81] in the beginning of the 1970s. In the following we outline these elementary complexity terms. Furthermore, we refer to the literature that shows that the problems we are concerned with are NP-hard.

2.3.1 Complexity Classes

In this section an intuitive overview of the complexity classes P, NP, NP-complete and NP-hard is given. These are concepts for a hierarchy of the complexity of problems on finite sets. The class with problems which can be solved in polynomial time is usually denoted by P. The class NP consists of those problems with the property that the correctness of a solution can be verified in polynomial time. It is not required that one is able to find the solution in polynomial time. The class NP includes most combinatorial optimization problems and all problems in P. NP is the shortcut for "solvable by a non-deterministic Turing machine in polynomial time". Within the complexity class NP the NP-complete problems are by definition the hardest problems. A problem is NP-complete if every problem in the class NP can be reformulated into this problem in polynomial time. Prominent problems which are known to be NP-complete are, for example, the traveling salesman problem, the knapsack problem, the maximum clique problem and the subgraph isomorphism problem. In figure 2.4 the discussed complexity classes are shown in a schematically diagram.

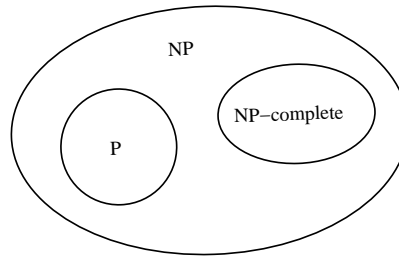


Figure 2.4: Schematically representation of the complexity classes P, NP and NP-complete

In the literature a problem L is called NP-complete if it is in NP and each problem in NP is Karp-reducible to L (see e.g. [3]). This implies, if one NP-complete problem is solvable in polynomial time, then each problem in NP can be solved in polynomial time. But although there is no proof, it is believed by most researchers that $P \neq NP$. This assumption might be supported by the fact, that most combinatorial optimization problems could either be proved to be polynomially solvable or to be NP-complete. For a precise analysis we refer to the books by Aho, Hopcroft and Ullman [1] and Garey and Johnson [50].

A definition for NP-hard problems can be found in [8]. When a decision version of a combinatorial optimization problem is proven to be in the class of NP-complete problems, then the optimization version of the problem is NP-hard. For example the decision version of the traveling salesman problem (TSP), "is there a tour with length less than r " is NP-complete. It is easy to verify that a proposed tour has a length less than r . But the optimization problem, "what is the shortest tour?", is NP-hard, since there is no easy way to determine if a given tour is the shortest.

2.3.2 Complexity of Discussed Graph Matching Approaches

The important characteristic of the NP-complete problems is that the run time required to solve problems in this class to optimality cannot be bounded from above by a polynomial that is a function of the problem size. This leads to the fact that one is only able to obtain guaranteed optimal solutions for small problem instances, for example, by exhaustive search. For larger instances one is obliged to use approximation algorithms which are only able to compute sub-optimal solutions.

Sahni and Gonzalez [124] showed in 1976 that for some problems, including the quadratic assignment problem, the corresponding ϵ -approximation¹ problem is also not polynomially solvable (If $P \neq NP$ is true.). The graph matching problem we are concerned with in chapter 4 is known to be NP-complete [50] as it can be stated as the quadratic assignment problem.

In chapter 5 we end up with an indefinite quadratic integer optimization problem which is also NP-hard [50] as Pardalos and Vavasis showed in [112] that indefinite quadratic programs are NP-hard problems, even if the quadratic program is very simple.

When dealing with approximations to the combinatorial graph matching problems one should keep in mind that the original problems are nearly impossible to solve in reasonable time.

2.4 Terminology and Mathematical Preliminaries

In the following we list the notation followed by some mathematical basics that are used throughout this thesis.

2.4.1 Notation

In this thesis the characters i, j, k and l are usually used as indices. Generally the lower case characters like x or e refer to column vectors. Capital letters like X, A and B refer to matrices. Greek letters like α, β and γ are used to represent scalar values.

Sets

\mathbb{R}^n	n-dimensional real vectors
$\mathbb{R}^{m \times n}$	$m \times n$ dimensional real matrices
\mathcal{S}^n	symmetric $n \times n$ matrices
\mathcal{S}_+^n	positive semidefinite $n \times n$ matrices: $\mathcal{S}_+^n = \{X \in \mathcal{S}^n X \succeq 0\}$
\mathcal{O}	orthogonal matrices X , i.e. $X^\top X = XX^\top = I_n$
\mathcal{E}	matrices with unit row and column sums
\mathcal{N}	non-negative matrices (i.e. non-negative elements)
Π	permutation matrices $\Pi = \mathcal{O} \cap \mathcal{E} \cap \mathcal{N}$
\mathcal{M}	matching matrices

¹A ϵ -approximate solution is a solution which has a fixed maximum deviation from the optimum.

Vectors and Matrices

e	vector of all ones: $e = (1, \dots, 1)^\top$
X^\top	transpose of the matrix X
I_n	$n \times n$ unit matrix
E_{nn}	$n \times n$ matrix of all ones: $E_{nn} = ee^\top$

Operators

$\text{Tr}[X]$	trace of the matrix X : $\text{Tr}[X] = \sum_{i=1}^n X_{ii}$
$X \bullet Y$	inner product of two matrices X, Y : $X \bullet Y = \text{Tr}[X^\top Y]$
$A \otimes B$	Kronecker product of A and B
$A \circ B$	Element wise Hadamard product: $(A \circ B)_{ij} = A_{ij}B_{ij}$
$\ X\ $	Frobenius norm of the matrix X : $\ X\ = \text{Tr}[X^\top X]^{1/2}$
$\text{vec}[X]$	vector obtained by stacking the columns of the matrix X
$\lambda(X)$	vector of the eigenvalues of the matrix X
$r(X)$	vector of the row sums of the matrix X
$s(X)$	sum of all elements of the matrix X
$\text{Diag}(x)$	diagonal matrix with the vector x as diagonal elements
$\text{diag}(X)$	vector of the diagonal elements of the matrix X

Miscellaneous

δ_{ij}	Kronecker delta: $\delta_{ij} = 1$ if $i = j$, and 0 otherwise
$A \succeq B$	<i>Löwner partial order</i> (see e.g. [99]): the matrix $A - B \succeq 0$ is positive semidefinite

2.4.2 Operator Definitions

Some of the above less common operator definitions which are used extensively within this thesis are sketched in more detail in the following:

Operator $\text{vec}[\dots]$

The operator $\text{vec}[X]$ creates a column vector $v = \text{vec}[X] \in \mathbb{R}^{mn}$ by appending the columns of the matrix $X \in \mathbb{R}^{m \times n}$ together. This is explicit shown for a 3×2 matrix resulting in a vector of dimension 6:

$$\text{vec}[X] = \text{vec} \left[\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix} \right] = (x_{11}, x_{21}, x_{31}, x_{12}, x_{22}, x_{32})^\top$$

Inner Product •

The inner product (scalar product), $A \bullet B$ of the matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times m}$ can be written as the trace of the matrix multiplication of A^\top and B .

$$A \bullet B = \text{Tr}[A^\top B]$$

For calculations the following property of the trace is very useful:

$$\text{Tr}[AB] = \sum_i^m \sum_j^n A_{ij} B_{ji} = \sum_j^n \sum_i^m B_{ji} A_{ij} = \text{Tr}[BA]$$

It says that the matrices within the argument of the trace can be cyclic exchanged.

Outer/Direct/Kronecker Product \otimes

The direct product $C = A \otimes B$, which is also called the Kronecker product, is the product of every element A_{ij} of the matrix $A \in \mathbb{R}^{n \times m}$ with the whole matrix $B \in \mathbb{R}^{p \times q}$ resulting in the larger matrix $C \in \mathbb{R}^{np \times mq}$. The elements of C are:

$$C_{\alpha\beta} = A_{ij} B_{kl}$$

with $\alpha = p(i-1) + k$ and $\beta = q(j-1) + l$. An illustrative example for $A \in \mathbb{R}^{2 \times 2}$ and $B \in \mathbb{R}^{3 \times 2}$ is shown here:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{11}b_{31} & a_{11}b_{32} & a_{12}b_{31} & a_{12}b_{32} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \\ a_{21}b_{31} & a_{21}b_{32} & a_{22}b_{31} & a_{22}b_{32} \end{pmatrix}$$

Some useful relations we occasional use in this thesis are:

$$\text{Tr}(AB) = \text{vec}(A^\top)^\top \text{vec}(B) \quad (2.1)$$

$$\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X) \quad (2.2)$$

$$(A \otimes B)(C \otimes D) = AC \otimes BD \quad (2.3)$$

These relations are easily verified by calculation. An extensive background related to the direct product can be found in the small specialized book “Kronecker Products and Matrix Calculus” written by Graham [56].

2.5 Basic Convexity Concepts

As we utilize convex optimization problems to approximate our graph matching problems we summarize a few basic facts about convex functions and sets. For more complete surveys along with several proves we refer to to

[20, 10, 109, 103, 106, 12] where especially in [20, chapter 2 and 3] convex sets and convex functions are discussed at great length. Convex relaxation and the important convex optimization problems which appear in this work are discussed in chapter 3.

2.5.1 Convex functions and sets

To ensure the attractive property that every local optimum is also a global optimum, convex optimization problems require, beside the convexity of the objective function, a convex domain. Therefore we recapitulate convex functions and convex sets and provide some examples of convex sets:

Convex functions

The real valued function $f(x)$, $x \in \mathbb{R}^n$ is convex if for any two points $x_1, x_2 \in \mathbb{R}^n$ the following condition is true:

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2) \quad \lambda \in [0, 1] \quad (2.4)$$

That means that the function values $f(x)$ on the line segment $x \in (\lambda x_1 + (1 - \lambda)x_2)$, $\lambda \in [0, 1]$ between any two points $x_1, x_2 \in \mathbb{R}^n$ are not above the line segment that is defined by the two end points $(x_1, f(x_1))$ and $(x_2, f(x_2))$. An example for a convex function is shown in figure 2.5.

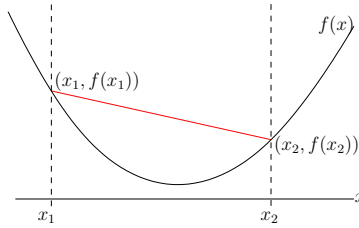


Figure 2.5: Example for a convex function: The function values $f(x)$ between any two points $x \in (\lambda x_1 + (1 - \lambda)x_2)$, $\lambda \in [0, 1]$ are below the corresponding line segment.

Convex sets

The set K is convex if for any two points x_1 and x_2 within the set K the line segment between the points is part of the set.

$$\lambda x_1 + (1 - \lambda)x_2 \in K \quad \forall \lambda \in [0, 1] \quad (2.5)$$

Figure 2.6 shows both an example for a 2D convex set and an example for a non-convex set.

Note that the solution set of a system of linear equations defines a convex set (see e.g. [20]).

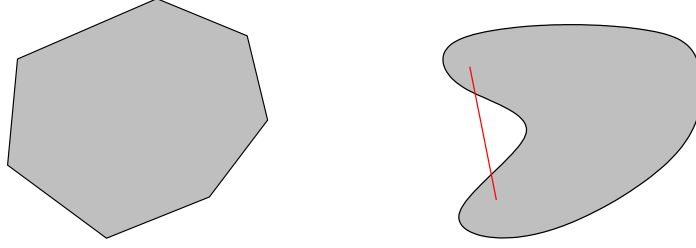


Figure 2.6: **Left:** Convex set **Right:** Non convex set (The shown line segment between two points of the set is not full contained in the set.)

Convex Cone

An important convex set called *cone* or *convex cone* is a set K which satisfies the following relations:

$$\begin{aligned} x + y &\in K \text{ for } x, y \in K \\ \lambda z &\in K \text{ for } z \in K, \lambda \geq 0 \end{aligned} \quad (2.6)$$

That means that a cone is a closed set under addition and multiplication with a positive scalar. The set of points in the nonnegative orthant and the set of positive semidefinite matrices are two known examples for cones.

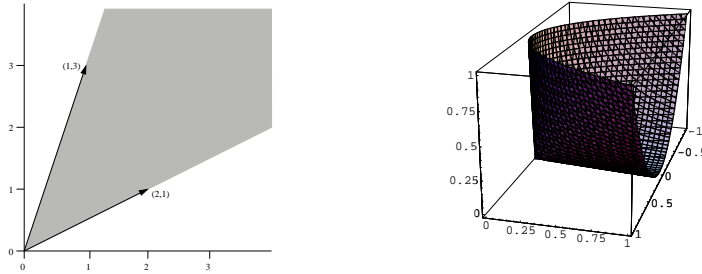


Figure 2.7: **Left:** Convex cone in R^2 **Right:** Surface of the positive semi-definite matrix cone of 2×2 symmetric matrices.

Recall that the set of positive semidefinite matrices \mathcal{S}_+^n is the set of symmetric $n \times n$ matrices which are positive semidefinite: $\mathcal{S}_+^n = \{X \in \mathcal{S}^n | X \succeq 0\}$ and that a positive semidefinite matrix can for example be characterized by the following property:

$$x^T Q x \geq 0 \text{ for all } x \in \mathbb{R}^n, Q \in \mathcal{S}_+^n$$

Figure 2.7 shows two examples of convex cones. The first is a cone in R^2 which can be represented as $K = \{\lambda_1(1,3)^T + \lambda_2(2,1)^T | \lambda_1, \lambda_2 \geq 0\}$. The second visualized the surface of the positive semi-definite matrix cone of 2×2 symmetric matrices

$$\begin{pmatrix} a & c \\ c & b \end{pmatrix} \in \mathcal{S}_+^2$$

Chapter 3

Combinatorial Relaxation and Convex Optimization

In this chapter we are concerned with methods of relaxations and some important facts about convex optimization. Together they build the basis of the convex relaxation approaches which are used to cope with the combinatorially hard graph and subgraph matching problems in chapter 4 and chapter 5. In section 3.1 we define what is meant by the term *relaxation* in this thesis and discuss why it represents an approximation of the original optimization problem. Then we outline how approximations can be obtained by Lagrangian Relaxation. Lagrangian duality and the convex optimization problems appearing within this thesis are the subject of section 3.2 and section 3.3. Appropriate methods to solve convex optimization problems are summarized in section 3.4. In section 3.5 we explain how we can efficiently compute a close feasible 0/1-integer solution to a real valued approximation of a matching.

3.1 Relaxations and Lower Bounds

Relaxation is a technique which can be used to obtain bounds to the optimum values of optimization problems. Moreover, it provides a method to compute approximate solutions for optimization problems which includes especially intractable combinatorial optimization problems. Note that a *convex relaxation* allows the efficient computation of the lower bounds.

Before turning to convex relaxation approaches we discuss the relaxation of general minimization problems where a relaxation leads to lower bounds. Relaxations are mainly based on a reasonable enlargement of the feasible set of an optimization problem, but can also involve a suitable change of the objective function. Consider the following general minimization problem where the function $f(x) \in \mathbb{R}$ has to be minimized over the set $x \in X$ which is assumed to be a subset of \hat{X} :

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X \subset \hat{X} \end{array} \quad (3.1)$$

A *relaxation* of the problem (3.1) can formally be written as

$$\begin{aligned} & \text{minimize} && \hat{f}(x) \\ & \text{subject to} && x \in \hat{X} \supset X \\ & && \hat{f}(x) \leq f(x) \quad \forall x \in X \end{aligned} \tag{3.2}$$

where the function $\hat{f}(x)$ is constrained not to be larger than $f(x)$ for any x on the original feasible domain X .

It is easy to see that the global solution of the relaxation (3.2) represents always a *lower bound* to the minimum of the original problem (3.1): if we assume that $\hat{f}(x) = f(x)$ for $x \in X$, then the complete original problem is embedded in the relaxation (3.2) which is minimized over the enlarged domain $x \in \hat{X} \supset X$. Therefore the minimum of the relaxed problem must be at least as low as the minimum searched over the original feasible set. Note that $\hat{f}(x)$ can be changed in an arbitrary way on $x \in \hat{X} \setminus X$. If the objective function $\hat{f}(x)$ is changed on $x \in X$ but the relation $\hat{f}(x) \leq f(x)$ on $x \in X$ is valid, then the relaxed solution can by definition not be higher as in the case where $\hat{f}(x) = f(x)$ for $x \in X$. Therefore the global solution of the relaxation (3.2) provides always a lower bound for the original problem (3.1). Note that this can be transferred straight forward to maximization problems, where the relaxation leads to an upper bound.

The calculation of lower bounds is often embedded into branch and bound frameworks for NP-hard optimization problems (see e.g. [21, 110, 60, 45]), but in the worst case these algorithms have exponential time complexity like their original counter-part. Therefore we decided in this work to utilize the global solutions of our relaxations as a direct approximation for the combinatorial solutions.

3.1.1 Interpretation of the Relaxation as Approximation

A natural and desired relaxation leaves the objective function on the original domain unchanged. That means that $\hat{f}(x) = f(x)$ holds for $x \in X$ in the relaxation (3.2) which is minimized over the enlarged set $x \in \hat{X} \supset X$. For combinatorial problems like our 0/1-graph matching problems the feasible domain of discrete decision variables $x \in \{0, 1\}^n$ is enlarged to the domain of real values $x \in \mathbb{R}^n$. If by chance, the optimal solution $x^* \in \mathbb{R}^n$ of the relaxed problem is also a feasible solution for the original problem ($x^* \in \{0, 1\}^n \subset \mathbb{R}^n$) then x^* must be the optimal solution for the original problem too. In such a case the relaxation approximates the original problem as close as possible. For slightly less tight relaxations it is likely that the relaxed solution x^* still shows the characteristics of the combinatorial solution.

This behavior and the fact that the original problem is embedded within the relaxation justifies the assumption that the solution x^* can be interpreted as an approximation for the integer solution of the original problem.

Unfortunately, the relaxed solution can not be expected to result in a feasible solution of the original problem but a post-processing step which computes a feasible integer solution close to the approximation makes it likely that the solution is close to the global combinatorial optimum. A measure for the accuracy of an approximation is therefore a measure which indicates how close the lower bound is to the combinatorial optimum. In our cases we utilize linear programs as discussed in section 3.5 to find appropriate 0/1-integer solutions close to the approximated solution.

3.1.2 Lagrangian Relaxation

A well known relaxation is the *Lagrange relaxation* which appeared first in 1970 in the today known form in the publications by Held and Karp [66, 67] where the authors proposed a branch and bound algorithm for the traveling salesman problem. Lagrange relaxation has its name since 1974 from Geoffrion [52] and provides a systematic way to compute an appropriate dual problem which bounds the given primal optimization problem. Consider the following general minimization problem where we assume $f(x)$, $g_i(x)$ and $h_j(x)$ to be real valued differentiable functions $\mathbb{R}^n \mapsto \mathbb{R}$:

$$\begin{aligned} & \min f(x) \\ & \text{subject to } g_i(x) = 0, \quad i = 1, \dots, l \\ & \quad h_j(x) \leq 0 \quad j = 1, \dots, m \\ & \quad x \in \mathbb{R}^n \end{aligned} \tag{3.3}$$

We refer to this problem as the *primal* program. The *Lagrange function* or *Lagrangian* $L(x, \lambda, \nu)$ which is associated to (3.3) is a function of the *primal* variable x and of the *dual* variables $\lambda \in \mathbb{R}^l$ and $\nu \in \mathbb{R}^m$ where $\lambda_i \neq 0$ and $\nu_i \geq 0$:

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^l \lambda_i g_i(x) + \sum_{j=1}^m \nu_j h_j(x) = f(x) + \lambda^\top g(x) + \nu^\top h(x) \tag{3.4}$$

The components λ_i and ν_j of the vectors λ and ν are called *Lagrangian multipliers*. The *dual function* is defined as the minimum of the Lagrange function over the primal variables and is thus a function of the dual variables λ and ν :

$$\Theta(\lambda, \nu) = \inf_{x \in \mathbb{R}^n} L(x, \lambda, \nu) \tag{3.5}$$

Note that the dual function (3.5) is concave in the dual variables, regardless of the convexity properties of $f(x)$ since it is the infimum of a family of concave functions (see e.g [20]). The non-positive term $\nu^\top h(x)$ guarantees that $L(x, \lambda, \nu)$ is lower than or equal to the original objective function over the original feasible set. Therefore the *Lagrangian relaxation* (3.5) indeed provides a lower bound to the minimization problem (3.3).

Finding the best and therefore largest lower bound (3.5) results in the following so called *dual problem* which is defined as the maximization of the dual function $\Theta(\lambda, \nu)$:

$$\sup_{\lambda \in \mathbb{R}^l, \nu \in \mathbb{R}^m} \Theta(\lambda, \nu) \quad (3.6)$$

Until now we did not make any convexity assumptions about the problem formulation which means that Lagrangian relaxation provides lower bounds also for non-convex problems.

3.1.3 Convex Relaxations

The purpose of relaxation methods is often to obtain a convex continuous optimization problem for otherwise intractable minimization problems. The convexity allows the efficient computation of the global solution of the relaxation which represents the approximation of the original problem.

Sometimes the convexity of a function is hidden in the original representation space and a transformation can be found to obtain an equivalent convex function (see e.g. [136]). A framework for convex relaxations of polynomial optimization problems over cones can be found in [89]. It covers a wide range of optimization problems such as linear and quadratic 0/1-integer programs, non-convex quadratic programs and bilinear matrix inequalities. In this thesis we are concerned with the following two approaches to obtain convex relaxations for graph and subgraph matching problems:

- In chapter 4 we utilize a non-trivial convex relaxation to a weighted graph matching approach. This particular convexification was obtained by Anstreicher and Brixius [6, 21] for quadratic assignment problems and results in a convex quadratic optimization problem.

In contrast to this particular convexification scheme the second convexification is based on a more generally applicable convexification scheme.

- Semidefinite programming has turned out to represent a powerful tool to approximate NP-hard binary optimization problems. In chapter 5 we use a widely applicable lifting scheme to approximate our subgraph matching approach by a (convex) semidefinite optimization problem.

In the following we shortly outline the optimality conditions for convex optimization problems and list the types of convex optimization problems we are concerned with in this thesis.

3.2 Optimality Conditions

The primal-dual pairs of optimization problems obtained by Lagrangian relaxation gives rise to the Lagrangian *duality theory* which provides the framework that lies behind many efficient algorithms for convex minimization problems. We present only the main results of the Lagrangian duality theory and refer for a thoroughly theoretical discussion to [20, 10, 94, 106].

3.2.1 Lagrangian Duality

Denoting the optimal value of the primal problem (3.3) with p^* and the optimal value of the dual problem (3.6) with d^* an important relation between these two is defined by the difference

$$p^* - d^*$$

which is called the *duality gap*.

Weak Duality

Due to the fact that the Lagrangian relaxation (3.6) provides a lower bound to (3.3) we have the following important inequality:

$$d^* \leq p^*$$

This property is called *weak duality* and guarantees a non-negative duality gap. Note that this relation is independent from any convexity assumption of the primal problem.

Strong Duality

If the duality gap is zero

$$d^* = p^*$$

then *strong duality* holds which means that the best bound that can be obtained is tight. In general strong duality does not hold but if the primal problem is convex there are mild conditions under which strong duality holds.

Slater's Condition

Slater's theorem says that strong duality holds if the primal program has a *strict feasible* point which means that there is a feasible point such that all inequalities are hold with strict inequalities. If the constraints are affine then strong duality holds under the weaker condition that a feasible point exists. A proof that strong duality holds when the primal problem is convex and Slater's condition holds can, for example, be found in [20, §5.3.2].

For convex optimization problems that satisfy Slater's condition, a certificate for optimality is a primal-dual pair of variables for which the Karush-Kuhn-Tucker-constraints hold.

3.2.2 Karush-Kuhn-Tucker Conditions

It is not so long since the optimality conditions for inequality constraint problems were published by Kuhn and Tucker [93] in 1951. They were disclosed first by Karush in his M.S. thesis [82] in 1939 and are nowadays referred to as *Karush-Kuhn-Tucker* (KKT) conditions.

Consider the minimization problem (3.3) with a differentiable object function and differentiable constraint functions along with the associated Lagrangian (3.4). A pair of primal and dual points x^* and (λ^*, ν^*) are *stationary points* if the following Karush-Kuhn-Tucker conditions are satisfied:

Primal feasibility:

$$g_i(x_s) = 0, \quad i = 1, \dots, l ; \quad h_j(x_s) \leq 0, \quad j = 1, \dots, m$$

Stationarity:

$$\nabla L(x, \lambda, \nu) = \nabla f(x)|_{x=x_s} + \sum_{i=1}^l \lambda_i \nabla g_i(x)|_{x=x_s} + \sum_{j=1}^m \nu_j \nabla h_j(x)|_{x=x_s} = 0 \quad (3.7)$$

Non-zero, Non-negativity:

$$\lambda_i \neq 0, \quad \nu_i \geq 0 \quad (3.8)$$

Complementarity:

$$\nu^\top h(x) = 0$$

For convex optimization problems the KKT conditions are sufficient optimality conditions. A proof can for example be found in [10].

3.3 Convex Optimization Problems

Due to the nice properties of convex optimization problems which allow the efficient computation of the global solution under mild conditions, our aim is to formulate the graph matching approximations and related problems as convex optimization problems.

3.3.1 General Convex optimization problems

A mathematical programming problem is called a *convex optimization problem* if it consists of a convex function $f(x)$ that is minimized on a convex domain X . Therefore, the optimization

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned} \quad (3.9)$$

is convex if $f(x)$ and X are convex. The function $f(x)$ is called the *objective function*. A point x is *feasible* if it is in X e.g. it satisfies the constraints that are used to define X . The point $x^* \in X$ is an *optimal solution* if $f(x^*) \leq f(x)$ for all $x \in X$.

3.3.2 Convex optimization problems in this thesis

In particular we are concerned with three types of convex optimization problems in this thesis: linear programming, quadratic programming and semidefinite programming. We shortly summarize the appearance of these programs within this work:

- Linear programming is utilized in this work in multiple ways. It appears in connection with the computation of maximal bipartite matching between two point sets. Very similar programs are used as post-processing to compute feasible 0/1-integer solutions from real valued approximations of the graph and subgraph matching problems. Furthermore linear programming is used within the computation of the so called *earth mover distance*, which is used to compute the similarity between grey-value-histograms of image parts.
- A convex quadratic program with linear constraints is the result of the non-trivial convex relaxation of a combinatorial weighted graph matching formulation in chapter 4.
- In chapter 5 the relaxation of a quadratic 0/1-integer program which models a subgraph matching approach results in a (convex) semidefinite programming problem.

These three types of convex optimization problems are summarized below along with their dual counter-parts which are derived by Lagrangian relaxation.

3.3.3 Linear Programming

The standard form of the linear programming (LP) problem is the following optimization problem:

$$\begin{aligned} \min \quad & q^\top x \\ \text{s. t.} \quad & Ax = c \\ & x \geq 0 \end{aligned} \tag{3.10}$$

Here the variables $q \in \mathbb{R}^n$ and $c \in \mathbb{R}^m$ are vectors and $A \in \mathbb{R}^{m \times n}$ is a matrix. Introducing the Lagrange multiplier u_i for the equality constraints and $s_i \geq 0$ for the inequality constraints the corresponding Lagrange function reads:

$$\begin{aligned} L(x, u, s) &= q^\top x - u^\top (Ax - c) - s^\top x \\ &= (q^\top - u^\top A - s^\top)x + u^\top c \end{aligned}$$

The dual function is

$$\Theta(u, s) = \min_x L(x, u, s) = \min_x ((q^\top - u^\top A - s^\top)x + u^\top c),$$

and from the stationarity conditions $\nabla_x L(x, u) = 0$ one finds a non-trivial minimum only if $(q^\top - u^\top A - s^\top) = 0$. Therefore we have the following dual function:

$$\Theta(u, s) = \begin{cases} u^T c & \text{if } (q^\top - u^\top A - s^\top) = 0 \\ -\infty & \text{otherwise} \end{cases}$$

Making the dual constraints explicit, we get the following known form of the dual problem:

$$\begin{aligned} \max \quad & u^T c \\ \text{s. t.} \quad & A^\top u + s = q \\ & s \geq 0 \end{aligned} \tag{3.11}$$

3.3.4 Quadratic Programming

The quadratic program with linear constraints we are concerned with in this thesis have the following form

$$\begin{aligned} \min \quad & \frac{1}{2} x^\top Q x + c^\top x \\ \text{s. t.} \quad & A x = b \\ & B x \leq d \\ & x \geq 0 \end{aligned} \tag{3.12}$$

where $x \in \mathbb{R}^n$ is a vector and $Q \in \mathbb{R}^{n \times n}$ is a matrix. The linear constraints are defined by the matrices $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times n}$ and the vectors $b \in \mathbb{R}^m$ and $d \in \mathbb{R}^p$. As the linear constraints already represent a convex set, the matrix Q has to be positive semidefinite to represent a convex optimization problem (see e.g. [106, §1.3.2]).

Without loss of generality we can assume that Q is a symmetric matrix, as the replacement of an unsymmetrical matrix \hat{Q} by the symmetric matrix $Q = \frac{1}{2}(\hat{Q} + \hat{Q}^\top)$ leaves the quadratic objective function unchanged. The corresponding Lagrangian ($v \geq 0, w \geq 0$) is

$$\begin{aligned} L(x, u, v, w) &= \frac{1}{2} x^\top Q x + c^\top x - u^\top (A x - b) - v^\top (d - B x) - w^\top x \\ &= \left(\frac{1}{2} x^\top Q + c^\top - u^\top A + v^\top B - w^\top \right) x + u^\top b - v^\top d, \end{aligned}$$

where $u \in \mathbb{R}^m, v \in \mathbb{R}^p$ and $w \in \mathbb{R}^n$ with $v, w \geq 0$. The stationarity condition

$$\nabla L(x, u, v, w) = x^\top Q + c^\top - u^\top A + v^\top B - w^\top = 0$$

lead us to the following dual function

$$\begin{aligned} \Theta(u) &= \min_{x \geq 0} L(x, u, v, w) \\ &= \begin{cases} u^\top b - v^\top d - \frac{1}{2} x^\top Q x & \text{if } A^\top u - B^\top v - Q x + w = c \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

and we obtain the dual problem by making the constraints explicit:

$$\begin{aligned}
\max \quad & u^T b - v^T d - \frac{1}{2} x^T Q x \\
\text{s. t.} \quad & A^T u - B^T v - Qx + w = c \\
& v, w \geq 0
\end{aligned} \tag{3.13}$$

3.3.5 Semidefinite Programming

The standard (primal) semidefinite program can be stated as

$$\begin{aligned}
\min \quad & \text{Tr}[QX] \\
\text{s.t.} \quad & \text{Tr}[A_i X] = c_i \quad \text{for } i = 1, \dots, m \\
& X \succeq 0,
\end{aligned} \tag{3.14}$$

where $Q, A_i \in \mathbb{R}^{n \times n}$ and $X \in \mathbb{R}^{n \times n}$ are matrices and $c_i \in \mathbb{R}, i = 1, \dots, m$ are scalar values. The matrices can be assumed to be symmetric matrices as every non symmetric matrix $\tilde{M} \in \mathbb{R}^{n \times n}$ can be made symmetric by $M = \frac{1}{2}(\tilde{M} + \tilde{M}^T)$ without changing the inner product $\text{Tr}[MX] = \text{Tr}[\tilde{M}X]$. Introducing the Lagrange multiplier $u_i \in \mathbb{R}, i = 1, \dots, m$ and $S \in \mathbb{R}^{n \times n}, S \succeq 0$, the Lagrange function $L(X, u, S)$ is:

$$\begin{aligned}
L(X, u, S) &= \text{Tr}[QX] - \sum_{i=1}^m u_i (\text{Tr}[A_i X] - c_i) - \text{Tr}[SX] \\
&= \text{Tr}[(Q - \sum_{i=1}^m u_i A_i - S)X] + c^T u
\end{aligned}$$

The stationarity condition

$$\nabla_X L(X, u, S) = Q - \sum_{i=1}^m u_i A_i - S = 0$$

reveals that the dual function is

$$\begin{aligned}
\Theta(u) &= \min_X L(X, u) = \min_X \text{Tr}[(Q - \sum_{i=1}^m u_i A_i - S)X] + c^T u \\
&= \begin{cases} c^T u & \text{if } Q - \sum_{i=1}^m u_i A_i - S = 0 \\ -\infty & \text{otherwise.} \end{cases}
\end{aligned}$$

Making the implicit constraints explicit, the dual semidefinite program reads:

$$\begin{aligned}
\max \quad & c^T u \\
\text{s.t.} \quad & Q - \sum_{i=1}^m u_i A_i - S = 0 \\
& S \succeq 0
\end{aligned} \tag{3.15}$$

The comparison of the primal-dual pair of semidefinite programs (3.14) and (3.15) with the linear programs (3.10) and (3.11) reveal that the semidefinite program can be seen as generalized linear programs where the non-negativity constraints in the linear program are replaced by semidefinite constraint on the matrix variables in the semidefinite programs.

Good survey articles for SDP include Vandenberghe and Boyd [140] the “Handbook of Semidefinite Programming” written by Wolkowicz, Saigal and Vandenberghe [146] and Helmberg [69].

3.4 Solving Convex Optimization Problems

Although the simplex method, invented by George Dantzig [35] in 1947, is efficient for most practical linear optimization problems, it was shown e.g. by Klee and Minty [88] that there exist cases where the simplex algorithm needs exponential time in terms of the problem size. Khachiyan’s ellipsoid algorithm [84] in 1979 was the first worst case polynomial time algorithm for linear programming, but for practical purposes it converges too slow. The publication of an efficient *interior point algorithm* for linear programming by Karmarkar in 1984 [80] has lead to the variety of now existing interior point algorithms.

The largest progress was made by extending interior point methods for linear programming to semidefinite programming which was proposed independently by Nesterov and Nemirovski [109] and Alizadeh [2] in 1994 and 1995.

3.4.1 Interior Point Methods

In recent years several interior point algorithms have been developed and the author of [48] divides interior point algorithms into three categories: affine scaling methods, potential reduction methods and central trajectory methods. The interior point methods based on the central trajectory are today the most used algorithms in practice and are also considered as most useful in theory. Therefore we sketch the main idea of these approaches: The basic idea of central path algorithms is to find a sequence of strictly primal-dual feasible points that converge along a central trajectory to the solution until the duality gap becomes smaller than a predefined threshold $\epsilon > 0$. Usually a barrier term with a weight $\mu > 0$ is added which keeps the points in the interior of the feasible region which is with increasing progress successively reduced. The most important property of interior point algorithms is that these algorithms represent worst case polynomial time algorithms.

3.4.2 Solver for Convex Optimization Problems

In this thesis we applied standard software to compute the arising convex optimization problems. Many solvers for convex linear, quadratic or positive semidefinite optimization problems are available freely or as commercial products.

- To solve the linear programs that appeared within this thesis we applied the CPLEX optimizer [76] which uses either a modified simplex method

or a primal-dual predictor corrector interior point method. An example for a freely available linear programming solver which is based on the simplex algorithm is LP-SOLVE [14].

- For the convex quadratic optimization problems which arise in chapter 4 we have used a freely available interior point solver which was developed by Zhang and Ye [148]. This interior point algorithm is designed to solve convex quadratic programs that are linearly constrained.
- To solve the semidefinite programs within this work we have applied the interior point solvers CSDP [18], PENSDP [92] and DSDP [13]. A benchmarking for this and several other SDP solvers can be found in [104].

3.5 Computing Integer Solutions for Assignments

The graph matching problems we are concerned with in chapter 4 and chapter 5 can be stated as combinatorial optimization problems. In particular they can be posed as optimization problems with 0/1-integer variables. A large fraction of such integer optimization problems are known to be NP-hard. Even general linear programs that underly the restriction that the feasible domain is integral are usually NP-complete problems [50]. As the approximations computed by the convex relaxations are usually not feasible for the original problem a post-processing step is required to obtain feasible 0/1 solutions. Luckily we can exploit some circumstances that allow the efficient computation of a feasible 0/1-integer solutions for our graph matching problems.

Integer Programming by Linear Programming

We can compute feasible 0/1-integer solutions which are close to real valued approximations of our graph matching problems by linear programming. This approach exploits the fact that feasible 0/1-integer solutions of our graph matching problems always represents a bipartite matching.

Computing an assignment (bipartite matching) represented by a matching matrix $X \in \mathcal{M}^{n \times m}$ which fits best to a real valued approximation $Q \in \mathbb{R}^{n \times m}$ of the matching matrix can be stated as the following integer optimization problem:

$$\max_{X \in \mathcal{M}} \text{Tr}[Q^\top X] = \max_{X \in \mathcal{M}} \sum_{i=1}^n \sum_{j=1}^m Q_{ij} X_{ij} \quad (3.16)$$

This problem maximizes the sum of the elements Q_{ij} that are “selected” by the matrix elements $X_{ij} = 1$ of the 0/1-matching matrix X . Recall from section 2.2.2, that the matching constraints for $X \in \mathcal{M}^{n \times m}, m \leq n$ are:

$$\begin{aligned} X_{ij} &\in \{0, 1\} \text{ for } i = 1, \dots, n, j = 1, \dots, m \\ \sum_{i=1}^n X_{ij} &= 1 \text{ for } j = 1, \dots, m \\ \sum_{j=1}^m X_{ij} &\leq 1 \text{ for } i = 1, \dots, n \end{aligned} \quad (3.17)$$

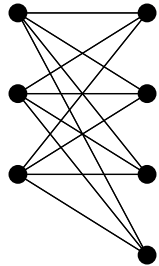
This means that all column sums of X are one while the row sum can be either 1 or 0. Note that in the case $n = m$ also the rows of X sums up to exactly 1 and X represents a permutation matrix. The interesting fact is that the problem (3.16) can be stated with $\text{Tr}[Q^\top X] = \text{vec}[Q]^\top \text{vec}[X] = q^\top x$ as the following linear program:

$$\begin{aligned} & \text{maximize} && q^\top x \\ & \text{subject to} && Ax = e \\ & && Bx \leq e \\ & && x \geq 0 \end{aligned} \tag{3.18}$$

Here the constraints reflect the row and column sum constraints of (3.17). The combined constraint matrices $A \in \mathbb{R}^{m \times nm}$ and $B \in \mathbb{R}^{n \times nm}$ turn out to represent a *totally unimodular* matrix $(B^\top, A^\top)^\top \in \mathbb{R}^{(n+m) \times nm}$. The totally unimodular matrix $(B^\top, A^\top)^\top$ along with the integer valued data e guarantees that the solution x^* of (3.18) is integral (see e.g. for proves [128, 91]) and results in a matching matrix if we interpret the solution vector $x^* = \text{vec}[X^*]$ as matrix X^* again. A definition of a totally unimodular matrix is given here:

- A matrix A is **totally unimodular** if every sub-determinant of A is 0, +1, or -1. A sub-determinant is the determinant of a squared sub-matrix of A which originates by deleting rows and columns of A . It follows that a totally unimodular matrix can have only elements with the values 0, +1, or -1.

For $n = 4$ and $m = 3$ the combined constraint matrix $(B^\top, A^\top)^\top$ looks like the matrix shown on the right side of figure 3.1. Seymour [130] showed that



$$(B^\top, A^\top)^\top = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3.1: Example for a bipartite graph with its incidence matrix. Incidence matrices of bipartite graphs are always totally unimodular.

all totally unimodular matrices can be constructed by so called network matrices (see e.g. [32]) and two further matrices that are totally unimodular. A polynomial-time total unimodularity test based on this result can be found in [128].

The fact that the matrix $(B^\top, A^\top)^\top$ is totally unimodular can be seen more easily as an incidence matrix of an undirected bipartite graph is always totally unimodular (see e.g. for a proof [91, Theorem 5.24]). An appropriate bipartite

graph that has an incidence matrix that is equal to the combined constraint matrix $(B^\top, A^\top)^\top$ for $n = 4$ and $m = 3$ is shown in figure 3.1.

Chapter 4

Weighted Graph Matching

In this chapter we study a specific part of an object recognition problem. In particular our aim is to find good matchings between relational structures that represent object views and which are assumed to be represented as weighted graphs. The discussed method is based on the observation that special weighted graph matching problems can be reformulated as quadratic assignment problems. We investigate the applicability of a recently developed convex quadratic relaxation approach for quadratic assignment problems to graph matching problems in pattern recognition.

4.1 Problem Statement

The primary purpose of this chapter is to investigate the applicability of a convex relaxation approach to inexact graph matching problems which may occur in visual object recognition systems. An example for a (weighted) graph representing an object view is shown in figure 4.1. The edge weights can be assumed



Figure 4.1: Image features and relations for an object view using features obtained with the FEX-system (cf. [47, 46]). The shown graph has $|V| = 38$ nodes.

to be any distance or similarity measure between adjacent feature vectors. The

aim is to find corresponding nodes between two view graphs based on the knowledge of the structure and the edge weights. It is well known that general graph matching problem are NP-hard (see section 2.3 or, e.g., [50]). For graphs like the one in figure 4.1, an exhaustive search has to check the impractical number of about $38! \approx 5 \cdot 10^{44}$ possible permutations of vertices. Therefore one has to resort to approximation algorithms to obtain good suboptimal solutions. The relaxation approach we are concerned with to tackle this problem was developed by Anstreicher and Brixius [5, 21] for the quadratic assignment problem (QAP). Before we define the graph matching problem in section 4.1.2 we briefly introduce the quadratic assignment problem in the following section.

4.1.1 The Quadratic Assignment Problem

The quadratic assignment problem can be interpreted as the problem to assign n factories to n places. The factories have to exchange a fixed amount of material. This results in a total cost which depends on the building cost, and a cost related to the exchanged material and the distance between the factories. The aim is to minimize the total cost. In figure 4.2 an example for a quadratic assignment problem is shown. The three factories should be placed on the three different places. Defining the flow matrix A , the distance matrix B and the cost matrix C for the building costs, the problem can be written as

$$\min_{X \in \Pi} f(X) = \min_{X \in \Pi} \text{Tr}[AXBX^\top + CX^\top]. \quad (4.1)$$

The objective function $f(X)$ has to be minimized over all possible assignments represented by the set of permutation matrices $\Pi \subset \mathbb{R}^{n \times n}$.

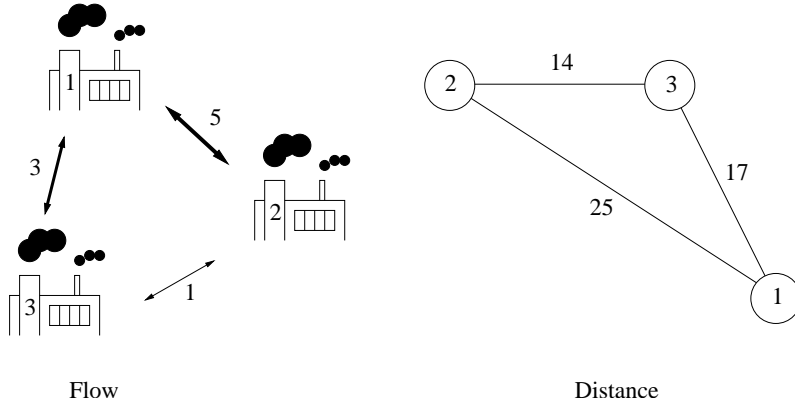


Figure 4.2: Small example of the quadratic assignment problem (QAP). The assignment of the factories to the places should minimize the total cost.

The QAP defines an important research field in combinatorial optimization. We refer to [24, 113] for surveys, and for more recent developments to [21, 120].

4.1.2 Graph Matching as QAP

We consider undirected weighted graphs $G = (V, E, w)$ with nodes $V = \{1, \dots, n\}$ and edges $E \subset V \times V$. The weight function $w : E \rightarrow \mathbb{R}_0^+$ encodes a similarity or

dissimilarity measure between pairs (i, j) of nodes. In image applications the nodes of the graph represent the features extracted from the image. The (dis-)similarity measure together with the structure of the graph is represented by the adjacency matrix A_G : $(A_G)_{ij} = w_{ij}$, $i, j = 1, \dots, n$. The adjacency matrices we are concerned with are symmetric $A_G^\top = A_G$, since the similarity measure is symmetric $w_{ij} = w_{ji}$.

Let $G = (V_G, E_G, w_G)$ and $H = (V_H, E_H, w_H)$ denote two given graphs with the same size $n = |V_G| = |V_H|$. An example for two similar graphs we would like to match is shown in figure 4.3. For illustration, the weights of the edges are proportional to the Euclidean distance between incident nodes.

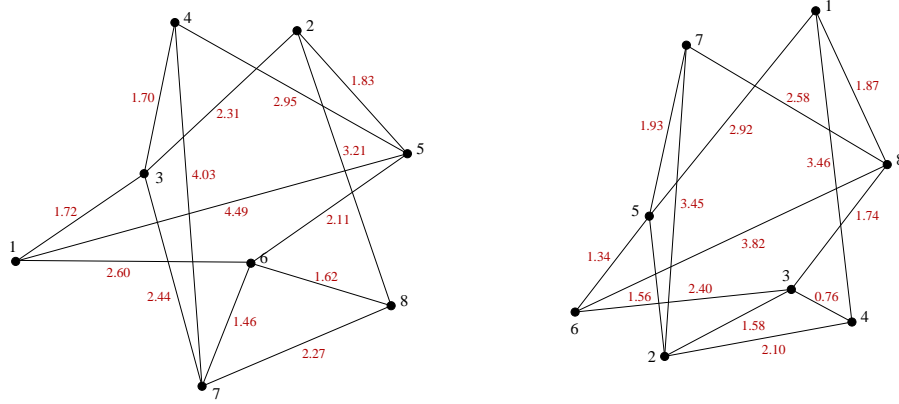


Figure 4.3: Example for two small graphs G and H with $|V_G| = |V_H| = 8$ to be matched. For illustration, the edge weights are proportional to the Euclidean distance between incident nodes.

In order to match these two graphs, we wish to compute a permutation $\Phi : V_G \mapsto V_H$ of the nodes of G such that the following distance measure is minimized:

$$\sum_{i,j=1}^n (w_{G,\Phi(i)\Phi(j)} - w_{H,ij})^2 \quad (4.2)$$

This measure is reasonable as for isomorphic graphs a permutation Φ exists such that the minimum value 0 is attained by mapping corresponding nodes to each other. Therefore it is likely to be an appropriate objective function if the two graphs are not too different. For such graphs the minimization of (4.2) prefers assignments which map edges with a similar edge weight to each other. In general, this measure favours mappings which assigns the edges with the largest weight to each other. The permutation Φ represents a mapping between the nodes of the first graph and the nodes of the second graph. Representing the permutation Φ by a permutation matrix $X \in \Pi$ (see section 2.2.2), the cost function (4.2) takes the following form in terms of the adjacency matrices of G and H [139]

$$f(X) = \|X A_G X^\top - A_H\|^2, \quad (4.3)$$

where $\|\cdot\|$ denotes the Frobenius norm.

As before, for isomorphic graphs the minimum value $f(X^*) = 0$ is attained by a mapping represented by a permutation matrix X^* which maps corresponding nodes to each other. For features V_G, V_H supplied by an image pre-processing step, it is unlikely that G and H are isomorphic. In this case we define as the best match the permutation matrix X^* which minimizes $f(X)$ over $X \in \Pi$. Thus, the graph matching problem formally reads:

$$f(X^*) = \min_{X \in \Pi} \|XA_GX^\top - A_H\|^2 \quad (4.4)$$

The minimization problem (4.4) is closely related to the quadratic assignment problem (QAP) introduced in section 4.1.1:

$$\min_{X \in \Pi} \text{Tr}[AXBX^\top + CX^\top] \quad (4.5)$$

This can be seen by reformulating the graph matching objective function as follows:

$$\begin{aligned} f(X) &= \|XA_GX^\top - A_H\|^2 = \text{Tr}[(XA_GX^\top - A_H)^2] \\ &= \text{Tr}[XA_GX^\top XA_G^\top X^\top] + \text{Tr}[A_HA_H^\top] - 2\text{Tr}[A_HXA_G^\top X^\top] \\ &= C_G + C_H - 2\text{Tr}[A_HXA_G^\top X^\top] \end{aligned} \quad (4.6)$$

Besides the term $C_H = \text{Tr}[A_HA_H^\top]$, the term $C_G = \text{Tr}[XA_GX^\top XA_G^\top X^\top]$ turns out to be constant in the case of equally sized graphs, as $XX^\top = X^\top X = I$ holds for permutation matrices. Therefore, we have $C_G = \text{Tr}[A_GA_G^\top]$. The assumption of equally sized graphs was also made by Umeyama in [139] where a spectral approach for weighted graph matching is described. The issue of extending the below explained technique to graphs of different size (*subgraph matching*) is discussed in section 4.8.

As a result, dropping the constant terms C_H and C_G , we recognize the graph matching problem (4.4) as a special case of the quadratic assignment problem (4.5) with $A = A_H$, $B = -A_G$ and $C = 0$. We therefore consider the following homogeneous quadratic assignment problem:

$$(QAP) \quad \min_{X \in \Pi} \text{Tr}[AXB^\top X^\top] \quad (4.7)$$

We note again that (4.2) corresponds to (4.7) only if $|V_G| = |V_H|$. We make this simplifying assumption in order to assess the convex relaxation techniques which have been developed for the quadratic assignment problem for the weighted graph matching problem in computer vision.

To demonstrate that the optimization problem (4.4), and therefore (4.7), is a reasonable graph matching approach, figure 4.4 depicts the matching obtained by the global minimum of (4.7) for the graphs shown in figure 4.3. The corresponding nodes are linked by red colored edges. It can be seen that the computed matching meets our expectation for this graph matching problem.

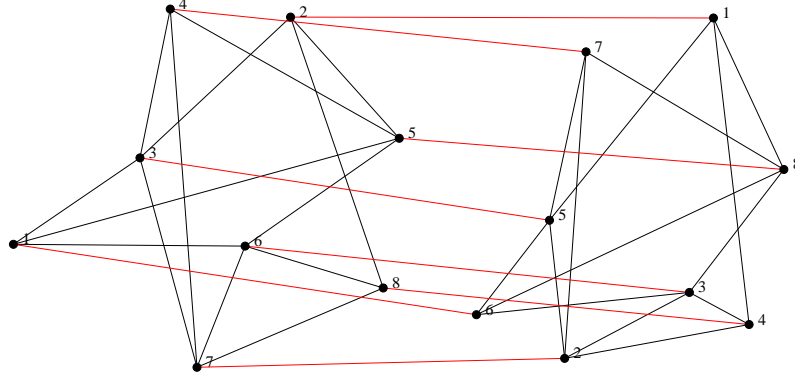


Figure 4.4: The matching obtained by solving the combinatorial minimization problem (4.4) or equivalently (4.7) for the graph matching example in figure 4.3. The found corresponding nodes are linked by red colored edges. The result is in accordance with the desired matching.

4.2 Relaxations and Lower Bounds

To approximate the combinatorial optimization problem (4.7) which is equivalent to the graph matching problem (4.4), several relaxations can be applied. In this section, we consider relaxations of (4.7) based on the constraints that characterize permutation matrices. To this end, we first characterize the properties of permutation matrices before we discuss three relaxation approaches in detail. We will see that an unique ranking of these approaches can be obtained based on the corresponding lower bounds.

4.2.1 Permutation Matrices and Relaxations

Possible natural relaxations for optimization problems which involve the set of 0/1-valued permutation matrices Π can be found by studying the properties which characterize this set.

To see which constraints can be used for relaxation we summarize the properties of permutation matrices $X \in \Pi$.

- Permutation matrices are orthogonal matrices: $X \in \mathcal{O}$

$$XX^\top = X^\top X = I$$

- As already discussed in section 2.2.2 permutation matrices have unit row and column sums, which is also known as the property that permutation matrices are *doubly stochastic*: $X \in \mathcal{E}$

$$Xe = X^\top e = e$$

- The elements of permutation matrices are non-negative: $X \in \mathcal{N}$

$$X_{ij} \geq 0 \quad \forall i, j$$

- Permutation matrices have 0/1-elements: $X \in \mathcal{Z}$

$$X_{ij}^2 - X_{ij} = 0 \quad \forall i, j$$

With these properties permutation matrices are completely characterized ([59, 149]):

$$\Pi = \mathcal{O} \cap \mathcal{E} \cap \mathcal{N} = \mathcal{E} \cap \mathcal{Z} = \mathcal{O} \cap \mathcal{Z}. \quad (4.8)$$

This gives rise to various relaxations. For example, if we drop the constraint that X has non-negative elements along with the doubly stochastic constraint we end up with the relaxation $X \in \mathcal{O} \supset \Pi$. This orthogonal relaxation is described in the following section. Next, the relaxation $X \in \mathcal{O} \cap \mathcal{E}$, which is based on a projection, is discussed. Finally the convex relaxation we are mainly interested in to obtain approximations for the graph matching problem, is the subject of section 4.2.4. This latter approach results in the tightest bound. In summary, using all constraints we can write the QAP as

$$\begin{aligned} \min_X \quad & \text{Tr}[AXB^\top X^\top] \\ \text{s.t.} \quad & XX^\top = X^\top X = I \\ & Xe = X^\top e = e \\ & X_{ij}^2 - X_{ij} = 0 \quad \forall i, j \\ & X_{ij} \geq 0 \quad \forall i, j, \end{aligned} \quad (4.9)$$

and reasonable relaxations for (4.9) are obtained by dropping one or more of the characterizing constraints. In the context of the Lagrange formalism (see section 3.1.2) the dropping of some constraints can be seen as Lagrangian relaxation with the corresponding Lagrange multipliers set to zero.

4.2.2 Orthogonal Relaxation

Relaxing the set of permutation matrices Π to the set of orthogonal matrices $\mathcal{O} \supset \Pi$, Finke et al. [44] suggested the so-called *Eigenvalue Bound* (EVB) which gives the following lower bound for the minimization problem (4.7):

$$(EVB) \quad \min_{X \in \mathcal{O}} \text{Tr}[AXB^\top X^\top] = \langle \lambda(A), \lambda(B) \rangle_- \quad (4.10)$$

Here, $\langle \lambda(A), \lambda(B) \rangle_-$ denotes the so-called *minimal scalar product*. This is the scalar product of the vectors $\lambda(A)$ and $\lambda(B)$ containing the eigenvalues of the adjacency matrices A and B ordered as follows: $\lambda_1(A) \leq \lambda_2(A) \leq \dots \leq \lambda_n(A)$ and $\lambda_1(B) \geq \lambda_2(B) \geq \dots \geq \lambda_n(B)$. The matrix X for which the bound (EVB) is attained can be calculated as well. If $U, W \in \mathcal{O}$ diagonalize the adjacency matrices A and B , respectively, and their columns are arranged according to the order of the eigenvalues mentioned above, then the minimum of (4.10) is attained at $X = UW^\top$. The advantage of this approach is that the matrix X can explicitly be calculated. However, it turned out that in many cases

this relaxation yields a bound for the minimization problem (4.7) which is too weak to be useful in practice. Furthermore it should be noted that $X = UW^\top$ is not an unique solution and that the minimum (4.10) is also attained for matrices $\tilde{X} = UDW^\top$ where D is a diagonal matrix with diagonal elements $D_{ii} \in \{1, -1\}$ (see Appendix A.1.2). This leads to 2^n possible solutions. To avoid this ambiguity, Umeyama used the matrices $|U|$ and $|W|$ in his graph matching approach [139] (which is based on the EVB) to obtain an approximate permutation matrix $X = |U||W|^\top$. Here $|U|$ and $|W|$ denote the matrices with elements that are the absolute values of the elements of the matrices U and W , respectively (cf. section 4.3.1).

4.2.3 Projected Eigenvalue Bound

Hadley et al. [59] improved the lower bound (4.10) by taking into account the doubly stochastic constraint $X \in \mathcal{E}$, in addition to the orthogonality constraint $X \in \mathcal{O}$. To this end, they parameterized relaxed permutation matrices $X \in \mathcal{O} \cap \mathcal{E}$ based on $(n-1) \times (n-1)$ orthogonal matrices $\hat{X} \in \mathcal{O}$ and the relationship

$$X = V\hat{X}V^\top + \frac{1}{n}E, \quad (4.11)$$

where $E = ee^\top$, and the $n-1$ columns of the $n \times (n-1)$ matrix V form a basis of the subspace orthogonal to the vector e . Conversely, for any $(n-1) \times (n-1)$ matrix $\hat{X} \in \mathcal{O}$, (4.11) yields a matrix $X \in \mathcal{O} \cap \mathcal{E}$. The $n \times (n-1)$ projection matrix V defined with any basis of $\{e^\perp\}$ can, for example, be calculated using the following scheme:

$$V = \begin{pmatrix} a & a & \cdots & a & a \\ 1+b & b & \cdots & b & b \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b & b & \cdots & b & 1+b \end{pmatrix} \text{ with } a = -\frac{1}{\sqrt{n}} \text{ and } b = -\frac{1}{n + \sqrt{n}}.$$

Using this parameterization, the objective function of the QAP can be rearranged as follows:

$$\text{Tr}[AXB^\top X^\top] = \text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top] + \text{Tr}[DX] - C_1, \quad (4.12)$$

where $\hat{A} = V^\top AV$, $\hat{B} = V^\top BV$, $D = \frac{2}{n}r(A)r(B)^\top$ and $C_1 = \frac{1}{n^2}s(A)s(B)$. The vector $r(A) = Ae$ denotes the vector of row sums of the matrix A , and the scalar $s(A) = e^\top Ae$ is the sum of all elements in A . The authors of [59] suggested to optimize the first two terms on the right hand side of (4.12) separately, the first one over $\hat{X} \in \mathcal{O}(n-1)$, and the second one over $X \in \Pi$. The latter problem amounts to solve the linear assignment problem (cf. section 3.5)

$$\text{LAP}(D) = \min_{X \in \Pi} \text{Tr}[DX], \quad (4.13)$$

which can be solved using any linear programming solver. As a result, the following lower bound for the minimization problem (4.7), the so called *Projected Eigenvalue Bound*, is obtained:

$$(PEVB) \quad \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + \text{LAP}(D) - C_1 \quad (4.14)$$

However, a major drawback of this bound is that due to the separate bounding of the two terms in (4.12), a corresponding minimizing matrix X , that can be used as approximate solution, cannot be computed in general. In [59] the authors discuss a special case in which it is possible to obtain a matrix X minimizing both terms. However, the next section shows how one can fully overcome this drawback by a convex relaxation approach and even achieve a better lower bound than (4.14).

4.2.4 Convex Relaxation

Following the work of Anstreicher, Brixius and Wolkowicz [5, 21], we focus on a convex relaxation of the minimization problem (4.7) in this section. The main motivation for this approach is its ability to compute *both* a tight lower bound and the corresponding matrix X where this bound is attained. The explicit calculation of X is important for us, as we use this result as direct approximation for the matching of the nodes of the two weighted graphs. In general, this is not possible for the bound (4.14) because it is obtained by minimizing two separate terms independently and hence does not lead to a single solution X .

As starting point for the convex relaxation, we reconsider the minimization of the first term of the right hand side of equation (4.12) over the orthogonal set \mathcal{O} of $(n-1) \times (n-1)$ matrices \hat{X} :

$$\begin{aligned} \min_{\hat{X}} \quad & \text{Tr}[\hat{A}\hat{X}\hat{B}^\top\hat{X}^\top] \\ \text{s.t.} \quad & \hat{X}\hat{X}^\top = I \\ & \hat{X}^\top\hat{X} = I \end{aligned} \tag{4.15}$$

The Lagrangian dual of this problem reads (see Appendix A.1)

$$\begin{aligned} \max_{\hat{S}, \hat{T}} \quad & \text{Tr}[\hat{S} + \hat{T}] \\ \text{s.t.} \quad & \hat{Q} := (\hat{B} \otimes \hat{A}) - (I \otimes \hat{S}) - (\hat{T} \otimes I) \succeq 0 \\ & \hat{S} = \hat{S}^\top, \hat{T} = \hat{T}^\top \end{aligned} \tag{4.16}$$

where \hat{S}, \hat{T} , like \hat{A} and \hat{B} , are $(n-1) \times (n-1)$ matrices. Here $\hat{Q} \succeq 0$ means that \hat{Q} has to be positive semidefinite. The solution for the dual problem (4.16) can be calculated by an equivalent linear program which can be derived using the fact that each of the two pairs of matrices \hat{A}, \hat{S} and \hat{B}, \hat{T} are simultaneously diagonalizable by \hat{U} and \hat{W} , respectively:

$$\begin{aligned} \max_{\hat{s}, \hat{t}} \quad & e^\top \hat{s} + e^\top \hat{t} \\ \text{s.t.} \quad & (\lambda_i \sigma_j - \hat{s}_j - \hat{t}_i) \geq 0 \quad i, j = 1, \dots, n \end{aligned} \tag{4.17}$$

Here σ_j and λ_i denote the eigenvalues of \hat{A} and \hat{B} , respectively. As \hat{s} and \hat{t} are vectors with the eigenvalues \hat{s}_j and \hat{t}_i of \hat{S} and \hat{T} , the matrices $\hat{S} = \hat{U}\text{Diag}(\hat{s})\hat{U}^\top$ and $\hat{T} = \hat{W}\text{Diag}(\hat{t})\hat{W}^\top$ can be obtained from (4.17) by linear programming. The

notation $\text{Diag}(\cdot)$ denotes the diagonal matrix with the vector in the argument on its diagonal. For details concerning the transformation of (4.16) to the linear program (4.17) we refer to the Appendix A.1.1.

The optimal solution for (4.15), according to (4.10), is

$$\min_{\hat{X} \in \mathcal{O}(n-1)} \text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top] = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- \quad (4.18)$$

The duality gap between the optimal solutions of (4.15) and (4.16) is zero since interior points exist for both problems (see, e.g., [147, 109] and [7] for an explicit proof). Hence, the optimal values are the same:

$$\max_{\hat{S}, \hat{T}} \text{Tr}[\hat{S} + \hat{T}] = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- \quad (4.19)$$

In order to exploit this relationship, we reformulate the objective function in (4.15) as follows:

$$\begin{aligned} \text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top] &= \text{vec}(\hat{X})^\top (\hat{B} \otimes \hat{A}) \text{vec}(\hat{X}) = \text{Tr}[(\hat{B} \otimes \hat{A}) \text{vec}(\hat{X}) \text{vec}(\hat{X})^\top] \\ &= \text{Tr}[(\hat{B} \otimes \hat{A}) Y] = (\hat{B} \otimes \hat{A}) \bullet Y, \end{aligned} \quad (4.20)$$

where we have used (2.1) and (2.2) and

$$Y = \text{vec}(\hat{X}) \text{vec}(\hat{X})^\top.$$

For arbitrary matrices \hat{S} and \hat{T} , and $\hat{X} \in \mathcal{O}$ the following equations hold:

$$\begin{aligned} \text{Tr}[\hat{S}] &= \text{Tr}[\hat{S}I] = \text{Tr}[\hat{S}\hat{X}\hat{X}^\top] = \text{Tr}[\hat{X}\hat{S}\hat{X}^\top] = \text{Tr}[I\hat{X}\hat{S}\hat{X}^\top] = (\hat{S} \otimes I) \bullet Y \\ \text{Tr}[\hat{T}] &= \text{Tr}[\hat{T}I] = \text{Tr}[\hat{T}\hat{X}I\hat{X}^\top] = (I \otimes \hat{T}) \bullet Y \end{aligned}$$

Here we have used $I = \hat{X}^\top \hat{X}$, the cyclic invariance of the trace, and again (2.1) and (2.2). Using these relations and assuming that \hat{S} and \hat{T} are feasible for the dual problem (4.16), a positive semidefinite quadratic form containing \hat{Q} from (4.16) can be introduced into the objective function $\text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top]$. To this end, the terms $\text{Tr}[\hat{S}] - (\hat{S} \otimes I) \bullet Y = 0$ and $\text{Tr}[\hat{T}] - (I \otimes \hat{T}) \bullet Y = 0$ are added to the objective function. Then the positive semidefinite term $\hat{Q} = (\hat{B} \otimes \hat{A}) - (I \otimes \hat{S}) - (\hat{T} \otimes I) \succeq 0$ can be identified. We get:

$$\begin{aligned} \text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top] &= (\hat{B} \otimes \hat{A}) \bullet Y \\ &= (\hat{B} \otimes \hat{A}) \bullet Y + \text{Tr}[\hat{S}] - (\hat{S} \otimes I) \bullet Y + \text{Tr}[\hat{T}] - (I \otimes \hat{T}) \bullet Y \\ &= \text{Tr}[\hat{S} + \hat{T}] + \underbrace{[(\hat{B} \otimes \hat{A}) - (I \otimes \hat{S}) - (\hat{T} \otimes I)] \bullet Y}_{\hat{Q} \succeq 0} \\ &= \text{Tr}[\hat{S} + \hat{T}] + \hat{Q} \bullet Y \\ &= \text{Tr}[\hat{S} + \hat{T}] + \text{vec}(\hat{X})^\top \hat{Q} \text{vec}(\hat{X}) \end{aligned}$$

Choosing \hat{S} and \hat{T} as the optimal solution to (4.16) we obtain with (4.19):

$$\text{Tr}[\hat{A}\hat{X}\hat{B}^\top \hat{X}^\top] = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + \text{vec}(\hat{X})^\top \hat{Q} \text{vec}(\hat{X})$$

Finally, substituting this expression as well as all the non-projected variables $\hat{X} = V^\top X V$ etc. into (4.12), we obtain after an elementary but tedious calculation (see also [22]) the quadratic formulation:

$$\text{Tr}[AXB^\top X^\top] = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + \text{vec}(X)^\top Q \text{vec}(X) \quad (4.21)$$

A comparison with (4.12) shows that now we just have a single term on the right hand side comprising the unknown matrix X . Hence (4.21) allows the computation of both a lower bound and the corresponding approximation matrix X . For the *linear* term in (4.12), minimizing over the set Π (cf. (4.13)) is equivalent to minimizing over $\mathcal{E} \cap \mathcal{N}$ (see 3.5). Accordingly, Anstreicher and Brixius [21] suggest to minimize the quadratic form in (4.21) over $\mathcal{E} \cap \mathcal{N}$, i.e. to solve the convex quadratic problem:

$$\begin{aligned} \min \quad & \text{vec}(X)^\top Q \text{vec}(X) \\ \text{s.t.} \quad & Xe = X^\top e = e \\ & X \geq 0 \end{aligned} \quad (4.22)$$

Here $X \geq 0$ means that all entries of the matrix X have to be non-negative. Using this minimization the *Quadratic Programming Bound* is given by:

$$(QPB) \quad \text{Tr}[AXB^\top X^\top] = \langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + \min_{X \in \mathcal{E} \cap \mathcal{N}} \text{vec}(X)^\top Q \text{vec}(X) \quad (4.23)$$

The following relationship between the bounds (4.10), (4.14) and (4.23) are hold by construction of the bounds (see also [21, 6]):

$$(EVB) \leq (PEVB) \leq (QPB) \leq (QAP) = \min_{X \in \Pi} \text{Tr}[AXB^\top X^\top] \quad (4.24)$$

Consequently the bound (4.23) computed by convex programming cannot perform worse than the other bounds. The quality of the corresponding solution X in comparison to other approaches (see next section) will be assessed in section 4.6.

4.2.5 Combinatorial Solutions

To obtain a permutation matrix $P \in \Pi$ from a non-integer solution $X \in \mathcal{E} \cap \mathcal{N}$ to (4.22), a good permutation matrix close to the approximation X has to be found. A simple way of doing this is to solve the following integer optimization problem which can be solved by linear programming (see also section 3.5):

$$P_0 = \arg \max_{P \in \Pi} \text{Tr}[X^\top P] \quad (4.25)$$

We prefer to use a slightly different approach which takes into account that in most cases a linear approximation of the original problem leads to an improvement of the obtained objective value. To this end, we add an unknown matrix Δ to the relaxed solution X so that one obtains a permutation matrix P :

$$P = (X + \Delta) \in \Pi$$

Next, we expand the objective function $\text{Tr}[APB^\top P^\top]$ around X up to linear terms with respect to Δ :

$$\begin{aligned}
\text{Tr}[APB^\top P^\top] &= \text{Tr}[A(X + \Delta)B^\top(X + \Delta)^\top] \\
&= \text{Tr}[AXB^\top X^\top] + \text{Tr}[AXB^\top \Delta^\top] + \text{Tr}[A\Delta B^\top X^\top] + \text{Tr}[A\Delta B^\top \Delta^\top] \\
&\approx \text{Tr}[AXB^\top X^\top] + \text{Tr}[AXB^\top \Delta^\top] + \text{Tr}[A\Delta B^\top X^\top] \\
&= -\text{Tr}[AXB^\top X^\top] + \text{Tr}[AXB^\top P^\top] + \text{Tr}[B^\top X^\top AP] \\
&= -\text{Tr}[AXB^\top X^\top] + 2\text{Tr}[B^\top X^\top AP]
\end{aligned}$$

As a result, we have to minimize the term $\text{Tr}[B^\top X^\top AP]$ to obtain the combinatorial solution P from the solution X of the relaxation. This problem can again be solved by linear programming:

$$P_1 = \arg \min_{P \in \Pi} \text{Tr}[B^\top X^\top AP]$$

To see the difference to (4.25), we put $M = -B^\top X^\top A$ and finally have:

$$P_1 = \arg \max_{P \in \Pi} \text{Tr}[MP] . \quad (4.26)$$

This minimization problem leads, compared to (4.25), in most cases to an improvement of the obtained objective value.

4.2.6 The 2opt Post-Processing Heuristics

A simple heuristic method called 2opt was proposed in [77] in order to further improve combinatorial solutions computed by more expensive methods. This greedy strategy iteratively exchanges pairs of assignments in the permutation until no further improvement is possible. It should be noted, however, that this is a completely local search strategy, in contrast to the non-local nature of convex relaxation. Yet, as a post-processing step of the latter, greedy search makes sense.

4.3 Other Approaches

In this section we briefly sketch two approaches that we will use for comparison with the convex relaxation approach of section 4.2.4. The first one was proposed by Umeyama [139] and resembles the spectral relaxation approach of [44]. Furthermore, we consider the deterministic annealing approaches [55] and [77] for which excellent performances are reported in the literature. The results concerning the annealing approaches (section 4.3.2) were calculated by Stefan Roth who investigated these approaches in his diploma thesis [122].

4.3.1 The Approach by Umeyama

Based on the *Eigenvalue Bound* (4.10), Umeyama [139] proposed the following estimate for the solution of (4.7):

$$X_{Ume} = \arg \max_{X \in \Pi} \text{Tr}(X^\top |U| |W|^\top) . \quad (4.27)$$

Here, U and W diagonalize the matrices A and B , respectively, with the eigenvalues sorted according to (EVB) , and $|\cdot|$ denotes the matrix consisting of the absolute values taken for each element. (4.27) is again a linear assignment problem which can be solved efficiently by using standard methods for linear programming (cf. section 3.5).

4.3.2 Graduated Assignment

Gold and Rangarajan [55] and Ishii and Sato [77] independently developed a technique commonly referred to as *graduated assignment* or *soft assign* algorithm. The set of permutation matrices Π is replaced by the convex set $\mathcal{D} = \mathcal{E} \cap \mathcal{N}$ of positive matrices with unit row and column sums (doubly stochastic matrices). In contrast to previous mean-field annealing approaches, the graduated assignment algorithm enforces hard constraints on row *and* column sums, making it usually superior to other deterministic annealing approaches. The core of the algorithm is an iteration scheme, which computes an approximate solution matrix X at each step of the annealing schedule. In our description $\beta > 0$ denotes the current annealing parameter; γ is a fixed “self-amplification” parameter, which enforces that the minimum on the set \mathcal{D} is also in Π . Denoting the iteration time step by the superscript, the matrix $X^{(r+1)}$ is calculated as follows (for fixed β):

$$X_{ij}^{(r+1)} = g_i h_j y_{ij}^{(r)} \quad (4.28)$$

Here $y_{ij}^{(r)}$ is

$$y_{ij}^{(r)} = \exp \left(-\beta \sum_{k,l} (A_{ik} B_{jl} + \delta_{ik} \delta_{jl} \gamma) X_{kl}^{(r)} \right).$$

The scaling coefficients g_i, h_j in (4.28) are computed so that $X^{(r+1)}$ is projected on the set \mathcal{D} using Sinkhorn’s algorithm [55] as inner loop. Stopping criteria based on convergence bounds or the number of iterations have to be established for the inner projection loop and the iteration scheme. For more details, we refer to [55, 77, 122].

Rangarajan et al. [118] showed that this scheme locally converges under mild assumptions. Several studies revealed excellent experimental results. In our experiments, we improved the obtained results with the local 2opt heuristics (cf. section 4.2.6).

A drawback of the graduated assignment algorithm is that the selection of several “tuning”-parameters is necessary to obtain optimal performance. An annealing schedule has to be set up, which is usually described by three parameters: an initial temperature, the annealing rate, and a final temperature or another stopping criterion [55]. There are theoretically motivated methods that give a lower bound for reasonable initial temperatures based on an analysis of the bifurcation structure of the problem [77]. Nevertheless, careful selection of the parameter greater than this bound can improve the results. The self-amplification parameter also has a lower bound that guarantees the above property that the minimizer of the objective function is in Π . An exhaustive parameter search for the annealing schedule, even below the theoretical bound,

may increase the performance. Finally, the stopping criteria also influence the quality of the results. All parameters have in common that their optimal values vary for different problem instances (cf. [77]), hence cannot be selected *apriori*.

4.3.3 SDP Relaxation

In this section we outline a SDP relaxation approach for QAPs. The reported bounds for QAPs obtained by semidefinite programming are generally better than the bounds obtained by the quadratic programming approach (see section 4.2.4). Recently published SDP bounds can be found in a paper by Rendl and Sotirov [120]. The first semidefinite programming approach was published by Zhao, Karisch, Rendl and Wolkowicz [149] who showed in detail that the dual of the Lagrangian dual of (4.9)¹ corresponds to an SDP relaxation of the QAP. The SDP relaxation bounds used here for comparison are taken from [120] when available.

We sketch the general semidefinite programming approach of [149] following the clearer presentation in [120]. Although we did not investigate this approach for graph matching we expect to obtain better approximations for the assignment by the (tighter) semidefinite programming relaxation. But due to the squared variables in the SDP approach the runtime of interior point algorithms for the quadratic optimization approach is significant lower. Some elements of this SDP approach for QAP are considered in more detail in chapter 5 where we explain our SDP subgraph matching approach.

To obtain the SDP relaxation, the QAP objective function is transformed by a Lagrangian relaxation which turns out to be equivalent to the lifting approach (see into a linear function (cf. 4.20))

$$\text{Tr}[AXB^\top X^\top + CX^\top] = (B \otimes A + \text{Diag}(c)) \bullet Y, \quad (4.29)$$

using $Y = \text{vec}(X)\text{vec}(X)^\top$ and the following reformulation of the second term on the left side of (4.29)

$$\text{Tr}[CX^\top] = c^\top \text{vec}(X) = c^\top (\text{vec}(X) \circ \text{vec}(X)) = \text{Tr}(\text{Diag}(c)Y) = \text{Diag}(c) \bullet Y,$$

where $c = \text{vec}(C)$. The matrix Y is positive semidefinite and has rank one. The condition $X_{ij}^2 - X_{ij} = 0$ for 0/1-constraints corresponds to

$$\text{diag}(Y) = y$$

where $y = \text{vec}(X)$ and $Y = yy^\top$.

Furthermore, the fact that the row and column elements of permutation matrices sum to one can be exploited. To this end, the projection matrix V

¹Using the more general objective function $\text{Tr}[AXB^\top X^\top + CX^\top]$ and without using the constraint $X_{ij} \geq 0 \quad \forall i, j$.

$$V = \begin{pmatrix} I_{n-1} \\ -e_{n-1}^\top \end{pmatrix}$$

known from section 4.2.3 – but here defined on a different basis of $\{e^\perp\}$ – is used to define a projection matrix W :

$$W = \left(\frac{1}{n}e \otimes e, V \otimes V\right)$$

Starting from $X = \frac{1}{n}E + V\hat{X}V^\top = \frac{1}{n}ee^\top + V\hat{X}V^\top$ (see section 4.2.3) and defining $\hat{x} = \text{vec}(\hat{X})$, we compute:

$$x = \text{vec}(X) = \frac{1}{n}(e \otimes e) + (V \otimes V)\hat{x} = W \begin{pmatrix} 1 \\ \hat{x} \end{pmatrix} = Wz ,$$

where $z = (1, \hat{x}^\top)^\top$. Hence, we obtain

$$Y = \text{vec}(X)\text{vec}(X)^\top = Wzz^\top W^\top = WRW^\top ,$$

where one can identify $R = zz^\top$ as a positive semidefinite matrix with rank one. Dropping the rank-one condition, the following semidefinite relaxation in terms of R can be defined

$$\begin{aligned} \min_R \quad & \text{Tr}[LR] \\ \text{s.t.} \quad & R_{11} = 1 \\ & \begin{pmatrix} 1 & y^\top \\ y & WRW^\top \end{pmatrix} \succeq 0 \\ & R \succeq 0 \end{aligned} \tag{4.30}$$

where $L = W^\top(B \otimes A + \text{Diag}(c))W$ and $y = \text{diag}(WRW^\top)$. The relaxation (4.30) turns out to be very weak in general. Fortunately, it can be tightened by the observation that the matrix $Y = \text{vec}(X)\text{vec}(X)^\top = WRW^\top$ has a structure that enforces many entries in Y to be zero if X is a permutation matrix $X \in \Pi$ (see [150]). To this end, the so-called *gangster operator* $G(WRW^\top) = 0$ is introduced in [149] to ensure that appropriate elements in WRW^\top are zero. In [120], non-negativity constraints on the entries of WRW^\top are used to further tighten the bound. The authors point out that the bundle method is an appropriate choice to deal with the high number of gangster and non-negativity constraints. Details on the SDP relaxation approach for the QAP can be found in the PhD-theses of Zhao [149, 150] and Sotirov [134].

4.4 Convex Relaxation: An Illustrative Numerical Example

For the purpose of illustration, we apply the convex relaxation approach from section 4.2.4 to a small graph matching problem in this section.

4.4.1 A Small Graph Matching Problem

In order to visualize the convex relaxation approach, we consider the two small weighted graphs G and H shown in figure 4.5. Obviously, the best match corresponds to exchanging vertices 2 and 3 in either graph.

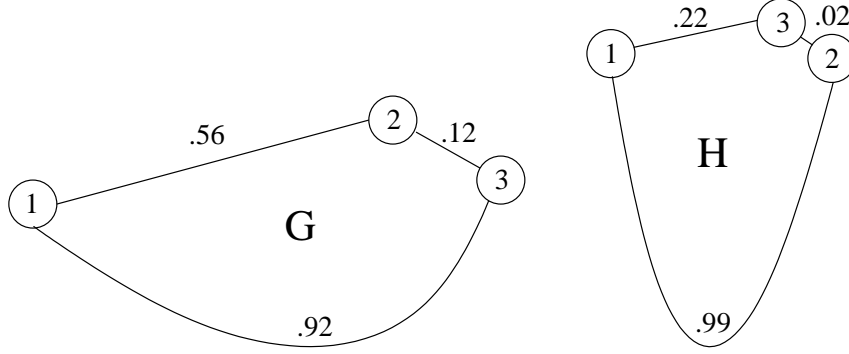


Figure 4.5: Two small sample graphs G and H to be matched

The adjacency matrices of the graphs G and H are:

$$A_G = \begin{pmatrix} 0 & 0.56 & 0.92 \\ 0.56 & 0 & 0.12 \\ 0.92 & 0.12 & 0 \end{pmatrix} \quad A_H = \begin{pmatrix} 0 & 0.99 & 0.22 \\ 0.99 & 0 & 0.02 \\ 0.22 & 0.02 & 0 \end{pmatrix}$$

In this example, the objective function of the graph matching problem (4.4) attains the following values for each of the six possible permutations:

$$1.370, 3.077, 2.01, 3.365, 0.613, 0.261$$

Thus, the optimum of this graph matching problem is

$$\text{OPT} = C_G + C_H + 2 \min_{X \in \Pi} \text{Tr}[AXB^\top X^\top] \approx 0.261$$

with $A = A_H, B = -A_G$, $C_G = \text{Tr}[A_G A_G^\top] \approx 2.349$ and $C_H = \text{Tr}[A_H A_H^\top] \approx 2.058$.

In the following, we visualize permutation matrices X by representing entries $X_{ij} = 1$ graphically by black squares, and $X_{ij} = 0$ by white squares. Accordingly, the permutation matrices which lead to the objective function values given above (in the same order) are depicted here:



The last permutation matrix represents an exchange of vertices 2 and 3 and thus corresponds to the global optimum of this graph matching problem.

4.4.2 Relaxations and Bounds

We calculate the bounds described in section 4.2 for the problem depicted in figure 4.5.

Orthogonal Relaxation

The eigenvalue bound (4.10) leads to the following lower bound of the optimal graph matching:

$$\text{EVB} = C_G + C_H + 2\langle\lambda(A), \lambda(B)\rangle_- \approx 0.023$$

with $C_G = \text{Tr}[A_G A_G^\top] \approx 2.349$, $C_H = \text{Tr}[A_H A_H^\top] \approx 2.058$ and $\langle\lambda(A), \lambda(B)\rangle_- \approx -2.192$. This bound is attained for $2^n = 2^3 = 8$ different matrices $X = UDW^\top$, where U, W diagonalize A and B , respectively, and D is a diagonal matrix with diagonal elements $D_{ii} \in \{1, -1\}$. Two possible solutions for X for which the bound is attained are:

$$X_1 \approx \begin{pmatrix} 0.999 & 0.041 & 0.002 \\ -0.014 & 0.316 & 0.948 \\ -0.038 & 0.948 & -0.317 \end{pmatrix} \quad X_2 \approx \begin{pmatrix} -0.027 & 0.529 & 0.848 \\ 0.963 & 0.240 & -0.119 \\ 0.267 & -0.814 & 0.517 \end{pmatrix}$$

Using linear programming to calculate permutation matrices close to these solutions, the first one gives the correct integer solution while the other one leads to a wrong integer solution. Besides the obvious weakness of the EVB-bound in general, this indicates that due to the ambiguity of the solution, it is unlikely to obtain a good approximation for the integer solution. The approach of Uneyama to take $X = |U||W|^\top$ leads to the following approximation, which in this case, yields the correct solution by linear programming:

$$X \approx \begin{pmatrix} 0.999 & 0.529 & 0.866 \\ 0.986 & 0.687 & 0.948 \\ 0.267 & 0.948 & 0.688 \end{pmatrix}.$$

Projected Eigenvalue Bound

Using the Projected Eigenvalue Bound (4.14), we obtain the following lower bound for our small graph matching problem:

$$\text{PEVB} = C_G + C_H + 2[\langle\lambda(\hat{A}), \lambda(\hat{B})\rangle_- + \text{LAP}(D) - C_1] \approx 0.181$$

where $\langle\lambda(\hat{A}), \lambda(\hat{B})\rangle_- \approx -0.985$, $\text{LAP}(D) \approx -2.003$, $C_1 \approx -0.875$. Note that this bound is much stronger than the EVB-bound. On the other hand, as mentioned in section 4.2.3, this approach does not allow to compute a corresponding matrix X for which the PEVB-bound is attained.

Quadratic Programming Bound

The Quadratic Programming Bound (4.23) gives:

$$\text{QPB} = C_G + C_H + 2[\langle\lambda(\hat{A}), \lambda(\hat{B})\rangle_- + \min_{X \in \mathcal{E} \cap \mathcal{W}} \text{vec}(X)^\top Q \text{vec}(X)] \approx 0.215$$

Here the minimization of the quadratic term results in $\min_{X \in \mathcal{E} \cap \mathcal{W}} \text{vec}(X)^\top Q \text{vec}(X) \approx -1.111$, and the bound is attained for

$$X \approx \begin{pmatrix} 0.747 & 0.000 & 0.253 \\ 0.253 & 0.000 & 0.747 \\ 0.000 & 1.000 & 0.000 \end{pmatrix}.$$

As predicted, this bound is superior to the PEVB-bound and leads to a good approximation of the correct permutation matrix. To summarize, for the numerical example considered here, the ranking (4.24) of these bounds reads as follows:

$$\text{EVB} \approx 0.023 \leq \text{PEVB} \approx 0.181 \leq \text{QPB} \approx 0.215 \leq \text{OPT} \approx 0.261 \quad (4.31)$$

4.4.3 Visualization

To illustrate how the convex relaxation approximates the original combinatorial problem, we graphically inspect the original objective function

$$f_{\text{orig}}(X) = C_G + C_H + 2\text{Tr}[AXB^\top X^\top]$$

along with its convex relaxation

$$f_{\text{convex}}(X) = C_G + C_H + 2[\langle \lambda(\hat{A}), \lambda(\hat{B}) \rangle_- + \text{vec}(X)^\top Q \text{vec}(X)]$$

for a few one-dimensional paths $X(\alpha)$ through the relaxed solution set defined by $X \in \mathcal{E} \cap \mathcal{N}$. It is well known (Birkhoff–von Neumann theorem) that this set is the convex hull of the original feasible set of the permutation matrices $X \in \Pi$. Hence, all paths

$$X(\alpha) = \alpha X_2 + (1 - \alpha)X_1, \quad \alpha \in [0, 1]$$

between extreme points $X_1, X_2 \in \Pi$ pass through the interior of the relaxed solution set, and we can graphically explore the two cost functions above by plotting their graphs over various paths. Figures 4.6 and 4.7 show several paths

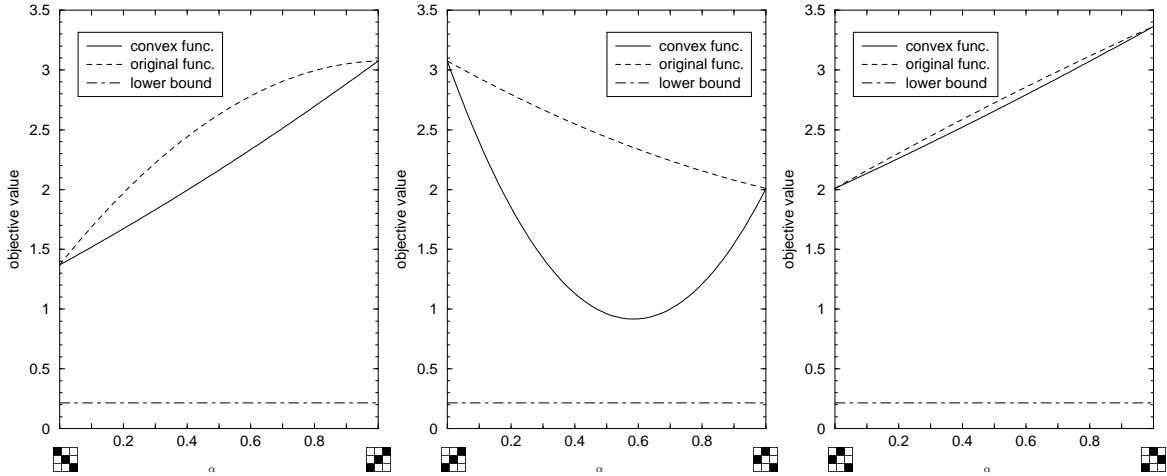


Figure 4.6: The original objective function and its convex relaxation along paths through the relaxed solution set.

through the relaxed set starting and ending at different permutation matrices. The following facts are illustrated:

- At the end-points of all paths, the two cost functions coincide because the convex relaxation approach does not change the original objective function on the original feasible set (cf. section 3.1).

- The original objective function is non-convex on the relaxed solution set and thus exhibits local minima. This is not the case for the objective function of the convex relaxation.
- The plot on the right hand side of figure 4.7 illustrates how the lower bound is attained by the convex relaxation approach. Furthermore, the point where this bound is attained is close to the global optimum (the end-point on the right), which confirms the tightness of the lower bound.

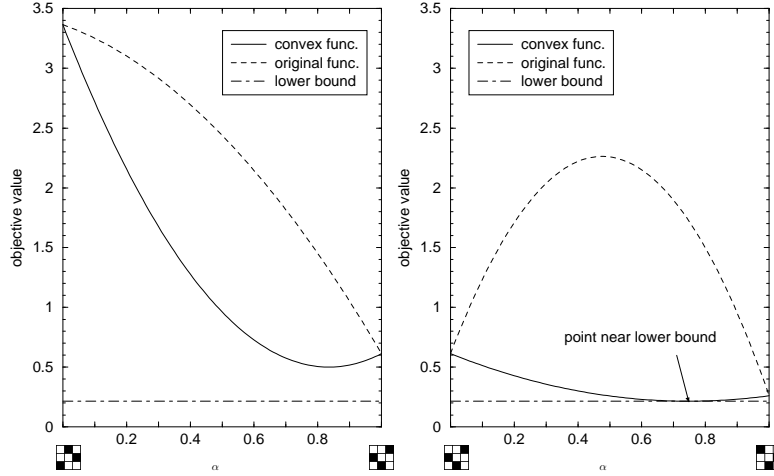


Figure 4.7: The original objective function and its convex relaxation along paths through the relaxed solution set. On the right, the plot illustrates how the lower bound is attained at a point close to the global optimum (end-point on the right).

In summary, these figures illustrate that the convex relaxation is tight and therefore “knows” where a “good” minimum is situated. Moreover, a point close to this global optimum can be computed without any initialization problem or parameter tuning.

4.5 Implementation Details

In this section we note some details concerning the implementation of the quadratic convex relaxation approach, which are not obvious in the first place. We used Mathematica [145] for the reformulation of the graph matching problem into the convex problem formulation (4.23). The quadratic program (4.22) itself was solved by a quadratic interior point solver developed by Zhang and Ye [148].

Our first remark concerns the quadratic solver. It turned out that the quadratic solver was able to solve our problems reliably only after a normalization of the problem data. We have chosen this normalization factor to be the maximum value of the absolute values in both matrices A and B . This results in normalized matrices with entries in the range $[-1, 1]$.

Secondly, despite the fact reported in [22], that the QPB (4.23) is not very sensitive to the choice of \hat{S} and \hat{T} obtained by the solutions \hat{s} and \hat{t} of the linear

optimization problem (4.17), we have chosen to implement a procedure called *update_basis* suggested in [22, §2.3] which updates the solution to \hat{s}' and \hat{t}' in order to potentially improve the calculated bound. We observed improvements of about 2.0 % with respect to the calculated bounds for the QAP problems with a rank deficit shown in table A.1 of the Appendix. For the problems shown in table 4.1, the improvement was negligible.

Thirdly, if \hat{A} or \hat{B} do not have full rank, ambiguous solutions of the linear optimization problem (4.17) are obtained. In this case we set the eigenvalues \hat{s}_j and \hat{t}_i to zero which correspond to zero eigenvalues $\sigma_j = 0$ and $\lambda_i = 0$, respectively.

4.6 Experiments

This section has three parts. In the first part, we investigate the performance of the convex relaxation. To this end, we compare the corresponding lower bounds with the combinatorial solutions of several benchmark problems from the QAPLIB-collection [25]. The QAPLIB is a public library of very difficult real-world quadratic assignment problems which can be used to evaluate and to compare the performance of any quadratic assignment approach. In the second part we present statistical results computed for a large set of randomly generated graphs (including ground-truth). Finally, in the third part a real world example is shown.

Abbreviations

The following abbreviations are used within the tables of this section. In general, f represents the value calculated for the objective function (4.7)

$$f(X) = \text{Tr}[AXB^\top X^\top],$$

with $X \in \Pi$. The subscript of f indicates which approach was used to obtain the solution $X \in \Pi$:

f^* :	value of the objective function (4.7) at the global optimum $X^* \in \Pi$
EVB :	the eigenvalue bound (4.10)
$PEVB$:	the projected eigenvalue bound (4.14)
QPB :	the quadratic programming bound (4.23)
f_{QPB} :	value of the objective function in (4.7) using the permutation matrix obtained with (4.25) from the QPB solution.
f_{QPB}^1 :	value of the objective function in (4.7) using the permutation matrix obtained with (4.26) from the QPB solution.
f_{GA} :	value of the objective function in (4.7) using the permutation matrix obtained by the graduated assignment algorithm (4.28).
f_{Ume} :	value of the objective function in (4.7) using the permutation matrix obtained by the approach of Umeyama (4.27).

An additional “+”-sign (e.g. f_{QPB+} , f_{QPB}^1+ , f_{GA+} , f_{Ume+}) indicates that the 2opt-heuristic method was used as a post-processing step to further improve the permutation matrix found.

4.6.1 QAPLIB Benchmark Experiments

Quality of the Relaxations

The quality of the relaxation approaches, namely the eigenvalue bound (EVB), the projected eigenvalue bound ($PEVB$), and the quadratic programming bound (QPB), can be assessed by measuring how close these bounds are to the global optimum (see (4.24)). Table 4.1 shows the results for problems drawn from

Problem	f^*	EVB	$PEVB$	QPB	SDP
chr12c	11156	-127514	-24375	-22648	n.a.
chr15a	9896	-190769	-52468	-48539	n.a.
chr15c	9504	-186403	-50295	-47409	n.a.
chr20b	2298	-30995	-8051	-7728	n.a.
chr22b	6194	-66432	-22126	-20995	n.a.
esc16b	292	-230	250	250	288
rou12	235528	-274122	200024	205461	223680
rou15	354210	-424419	296705	303487	333287
rou20	725522	-739730	597045	607362	663833
tail0a	135028	-181950	112528	116260	n.a.
tail2a	224416	-284261	193124	199378	219760
tail5a	388214	-414351	325019	330205	358802
tail7a	491812	-496403	408910	415578	451317
tail20a	703482	-714901	575831	584942	637300
tai30a	1818146	-1505553	1500406	1517829	1652186
tai35a	2422002	-2015233	1941622	1958998	n.a.
tai40a	3139370	-2559063	2484371	2506806	n.a.

Table 4.1: Bounds computed for QAPLIB-problems.

the QAPLIB [25]. The first column comprises labels indicating the problem and the number $|V|$ of vertices of a data set from the QAPLIB. The second column shows the value of the objective function at the global optimum. The lower bounds $EVB, PEVB$ and QPB computed by the corresponding relaxation approaches are listed in the next three columns. The last column shows the available bounds obtained by the SDP relaxation sketched in section 4.3.3, and are taken from [120].

Since zero is a trivial lower bound for the QAPLIB problems, a negative sign indicates that the relaxation is not at all tight. As this happens for all problems with the bound EVB , it can be considered not to be useful. Note that the first five problems in table 4.1 seem to be the most difficult, as all the computed bounds are negative².

Furthermore, table 4.1 confirms the relationship (4.24), with the quadratic convex relaxation approach giving a good lower bound. The fact that the available bounds obtained by the SDP relaxation represent the best bounds supports the assumption that SDP relaxations are very tight relaxations. With ongoing development of SDP solvers and an improved capability to cope with more and more constraints, we believe that the SDP relaxation will become the method of choice to obtain approximations for QAPs.

²It would be interesting to know the corresponding SDP results.

Comparison to Spectral Decomposition and Graduated Assignment

We compare the combinatorial solutions obtained with the convex relaxation approach with those computed with the graduated assignment approach (section 4.3.2) and the spectral decomposition approach by Umeyama (section 4.3). Table 4.2 shows the corresponding results in the same way as table 4.1. A “+”–

Problem	f^*	f_{QPB}	f_{QPB+}	f_{QPB}^1	f_{QPB+}^1	f_{GA}	f_{GA+}	f_{Ume}	f_{Ume+}
chr12c	11156	20306	15860	27912	13088	19014	11186	40370	11798
chr15a	9896	26132	14454	20640	13540	30370	11062	60986	17390
chr15c	9504	29862	17342	19436	12754	23686	13342	76318	13338
chr20b	2298	6674	2858	7276	3832	6290	2650	10022	3294
chr22b	6194	9942	6848	8958	6902	9658	6732	13118	7418
esc16b	292	296	292	312	292	298	292	306	292
rou12	235528	278834	246712	266864	241802	273438	246282	295752	251848
rou15	354210	381016	371480	394192	374000	457908	359748	480352	384018
rou20	725522	804676	746636	795578	757270	840120	738618	905246	765872
tail0a	135028	165364	143260	154282	139524	168096	135828	189852	147838
tail2a	224416	263978	237200	246424	238902	263778	224416	294320	252044
tail5a	388214	455778	399732	432610	390782	451164	400328	483596	405442
tail7a	491812	550852	513170	545410	526518	589814	505856	620964	526814
tail20a	703482	799790	740696	752896	726038	871480	724188	915144	775456
tail30a	1818146	1996442	1883810	1979530	1872722	2077958	1886790	2213846	1875680
tail35a	2422002	2720986	2527684	2677688	2511800	2803456	2496524	2925390	2544536
tail40a	3139370	3529402	3243018	3411278	3277450	3668044	3249924	3727478	3282284

Table 4.2: Results of the QAPLIB benchmark experiments (see text).

sign indicates that the 2opt–heuristic method was used as a post–processing step to improve the solution. The difference between f_{QPB} and f_{QPB}^1 is that linearization was used to “round” the convex programming solution to a combinatorial solution in the latter case (see section 4.2.5).

The columns labeled with f_{GA} and f_{Ume} show the results obtained with the graduated assignment approach [55, 77] and with the approach by Umeyama [139]. It should be noted that considerable care was taken to manually find out optimal parameter values for the graduated assignment approach for *each* data set [122].

The following conclusions can be drawn from the results shown in table 4.2:

- The convex relaxation approach f_{QPB} and the soft-assign approach f_{GA} have similarly good performance. However, the latter approach is much more intricate from the optimization point-of-view and involves a couple of tuning parameters which had to be optimized by hand.
- The approach of Umeyama f_{Ume} based on spectral decomposition is less competitive.
- Using the simple 2opt greedy–strategy as a post–processing step significantly improves the solutions.

4.6.2 Random Ground-Truth Experiments

In this subsection, we discuss results obtained for two different ground-truth experiments. In the first experiment, we created many problem instances (4.7) by independently computing two different random graphs with the same number of vertices. In the second experiment we computed a large collection of random

graphs along with slightly perturbed and randomly permuted “copies” of these graphs. The experiment is supposed to simulate a vision system that obtains object graphs which are slightly perturbed due to distortion or noise.

Random Graphs

In this experiment, we created many problem instances (4.7) by independently computing two different random graphs with the same number of vertices. The probability that an edge is present in the underlying complete graph was 0.3. Figure 4.8 shows an example in order to visualize the edge-density of such graphs. The global optimum for (4.7) was computed with an exact search algorithm. This optimum was then used to calculate the ratio between the sub-optimal objective value and the best objective value for each problem instance. Hence, the best ratio possible is 1.0 if the lower bound coincides with the best objective value. Table 4.3 summarizes the results by showing the statistics (mean, worst case, and best case) for three experiments with different sizes of the graphs ($n = 9, 11, 15$). The number of problem instances for each experiment is shown in angular brackets. The number of correctly found matchings without/with the 2opt heuristic method as post-processing step are shown in round brackets.

The following conclusions can be drawn from the results shown in table 4.3:

- The soft-assign approach performs somewhat better for these experiments than the convex relaxation approach, which yet does involve no tuning parameters that have to be optimized by hand and does not depend on initialization.
- With increasing problem size the performance decreases for all three approaches in general.
- The approach of Umeyama f_{Ume} based on spectral decomposition is less competitive.
- Using the simple 2opt greedy-strategy as a post-processing step significantly improves the results.

	f_{QPB}^1/f^*			f_{Ume}/f^*			f_{GA}/f^*		
	mean	worst case	best case	mean	worst case	best case	mean	worst case	best case
$n=9$ [128]	(22/55)			(7/29)			(31/55)		
	0.88765	0.43810	1	0.638244	0.065173	1	.948342	.7756129	1
+2opt	0.97130	0.79256	1	0.928304	0.753007	1	.969914	.843046	1
$n=11$ [42]	(3/10)			(0/7)			(7/10)		
	0.83043	0.56268	1	0.636159	0.295194	0.998591	.940740	.8338586	1
+2opt	0.95760	0.85043	1	0.933206	0.811326	1	.958863	.8434407	1
$n=15$ [99]	(0/2)			(0/1)			(4/11)		
	0.78726	0.52307	0.938917	0.225983	0.131333	0.863508	.916225	.105164	1
+2opt	0.92195	0.77956	1	0.890131	0.74688	1	.95763	.820596	1

Table 4.3: Statistics of the results of random ground-truth experiments (see text).

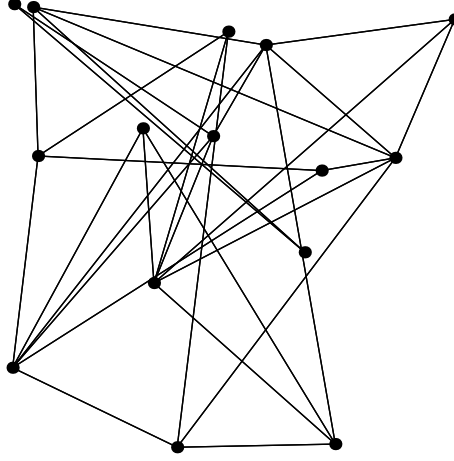


Figure 4.8: A randomly generated graph with 15 vertices and a probability of 0.3 for the presence of an edge.

Perturbed Graphs

In the second series of experiments, we computed a large collection of random graphs along with slightly perturbed and randomly permuted “copies” of these graphs. The weights of the second graph were perturbed by a normally distributed factor with standard deviation $\sigma = 0.1$ and means 1. Using the same structure as table 4.3, the corresponding results are shown in table 4.4. For larger problems (more than 15 vertices) for which computing the global optimum was too expensive, we assumed the optimal permutation to be the inverse of the random permutation matrix that was used to compute the second graph of each pair. In some cases this was not true, however, and hence a different permutation with a lower objective value could be found by the algorithms. This explains why some of the quotients in table 4 have a value greater than 1.

In summary, the statistics of our results shown in table 4.4 reveal that in almost every case of these “low-level noise” experiments the optimal permutation was found by the quadratic programming approach. In contrast, the other approaches show a slightly decreasing performance with an increasing problem size.

	f_{QPB}^1/f^*			f_{Ume}/f^*			f_{GA}/f^*		
	mean	worst case	best case	mean	worst case	best case	mean	worst case	best case
n=9 [155]		(154/155)			(142/154)			(144/151)	
2opt	0.999996	0.999382	1	0.986481	0.463282	1	.997206	.859380	1
	1	1	1	0.999883	0.981862	1	.998154	.859380	1
n=15 [183]		(183/183)			(163/175)			(176/181)	
2opt	1	1	1	0.974078	0.379189	1	.998347	.833787	1
	1	1	1	0.993836	0.718871	1	.998484	.833787	1
n=20 [173]		(173/173)			(148/163)			(167/171)	
2opt	1	1	1	0.977225	0.475711	1	.998205	.855257	1
	1	1	1	0.991662	0.772512	1	.998338	.855257	1
n=25 [169]		(169/169)			(64/123)			(126/143)	
2opt	1.00001	1	1.00155	0.848105	0.216079	1	.966097	.491432	1.001550
	1.00002	1	1.00155	0.960519	0.602629	1.00099	.9748815	.686842	1.001550

Table 4.4: Statistics of the results of perturbed graph experiments (see text).

4.6.3 Real World Example

In this section, we show an example for a real world graph matching problem where the nodes of the object graphs are based on features that can be found by an appropriate feature extractor like, for example, the FEX-system (cf. [47, 46]). The graphs we want to match are shown in figure 4.9. They have 38 nodes, which means that there is the tremendous number of approximately 10^{44} possible assignments.

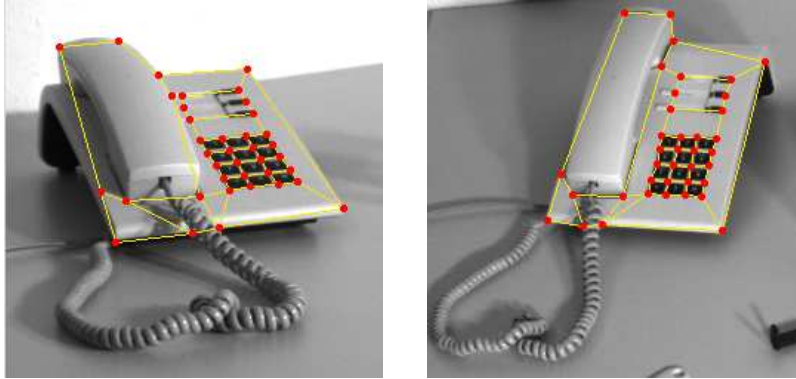


Figure 4.9: Two graphs representing the same object. The graphs are created based on features obtained with the FEX-system (cf. [47, 46]).

The result of the graph matching experiment is shown in figure 4.10. The convex relaxation was able to find the expected assignment which is very encouraging because the number of possible assignments is very huge. It should also be mentioned that the results shown in this section are not further optimized by the 2opt post-processing step and are therefore the results of the convex relaxation approach (4.23), together with the linear optimization (4.26) which is a convex problem too. This first experiment resembles the series of “low-level noise” experiments from the previous section as only the nodes are permuted and the weights of the graph are perturbed, but the underlying structure is preserved.

To show that the graph matching approach is also applicable in situations where in addition to the edge weights also the structure of the graphs is disturbed, we conducted a second experiment. This experiment is shown in figure 4.11 where the additional perturbations in the underlying structure are marked by green edges. The result gets nearly the desired matching where the undesired matching occurs surprisingly in the left bottom part of the graphs. The undesired matchings are represented by blue line segments.³

These real world examples in connection with the comprehensive statistical results reported above show that the convex relaxation approach is appropriate to find the desired matching for similar graphs of the same size ($|V_G| = |V_H|$) which do not differ too much. In the next section we discuss the advantages and disadvantages that are inherent to this graph matching approach.

³This error is likely to be adjusted by the 2opt heuristics (see section 4.2.6).

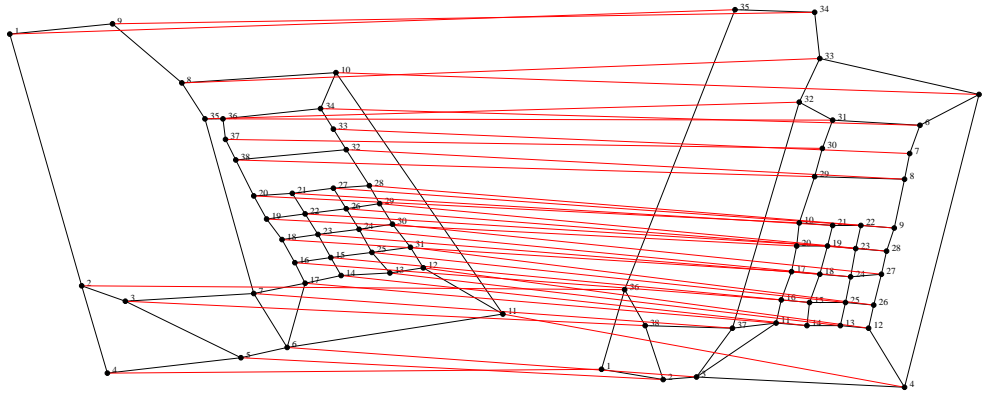


Figure 4.10: For the shown object graphs the desired matching is obtained by the convex optimization approach.

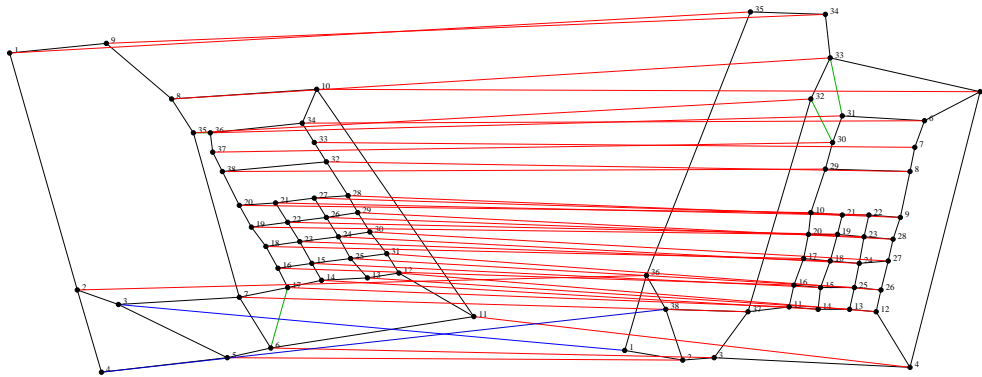


Figure 4.11: Matching result obtained with the convex optimization approach. In the left graph also the structure of the graph is perturbed. The perturbations in the structure are marked with green edges. Nearly the desired matching is found. Undesired matchings are represented by blue line segments.

4.7 Discussion

To reveal the advantages but also the limits of the QAP graph matching approach we point out its inherent properties in this section. We start with identifying the invariants for our approach. Then we discuss conditions in which this approach does not lead to a desired matching.

4.7.1 Invariants

To find invariants we investigate which transformations applied to the graphs leave the results obtained by the graph matching approach unchanged. In particular, we find that the approach is invariant against rotation, translation and scaling of the objects. These three invariants are discussed below:

Rotation

The graph matching approach is invariant against an arbitrary 2D rotation of any of the two objects. The reason for this lies in the fact that the adjacency matrices, from which the assignments are calculated, are only dependent on the pairwise relationship (distance) of the nodes which does not change under a rotation of an object in an image. That means the adjacency matrix A_G of the object graph G is equal to the adjacency matrix $A_{G'}$ of the rotated object graph G' .

Translation

Clearly, the same reasoning as for the rotation leads to the statement that the approach is also invariant against any translation of the objects. The adjacency matrices are not changed by a translation of either of the object graphs obtained from an image.

Scaling

Not that obvious is the invariance of the QAP graph matching approach against scaling of any of the two objects. This can be seen by considering the following reformulation of the optimization problem (cf. (4.7))

$$\min_{X \in \Pi} \text{Tr}[AXB^\top X^\top] = \min_{X \in \Pi} s \text{Tr}\left[\frac{A}{s}XB^\top X^\top\right] = s \min_{X \in \Pi} \text{Tr}[\tilde{A}XB^\top X^\top] \quad (4.32)$$

where we have introduced a scalar scaling factor $s > 0$ and the scaled matrix $\tilde{A} = \frac{A}{s}$. The optimal permutation does not change if we omit the constant scaling factor s on the right hand side of (4.32) and therefore the optimal solution $X^* \in \Pi$ is independent from a scaling of the matrix A . As the same is valid for a scaling of B , the optimal permutation does not change if either of the two graphs is scaled.

4.7.2 Limitations of the QAP Graph Matching

In this section we reveal cases in which the QAP graph matching approach is likely to lead to undesired matchings. In particular, we discuss a case where the objective function appears not to be appropriate to model the desired graph matching.

Structural Perturbations

The real world example from section 4.6.3 has shown that the QAP graph matching approach is non-sensitive against small perturbations. However, it turns out that larger perturbations more likely lead to undesired matchings. This behavior is due to the fact that the minimization of the objective function in (4.7) prefers matchings where edges with large weight of the first graph are mapped to edges with large weight in the second graph. This can be seen for example for the distance measure (4.2) that is minimized by the QAP graph matching approach: The strongest gains are obtained if edges with a large weight are mapped to each other. On the other hand, this makes sense from the viewpoint of computer vision, since large weights can be expected to involve reliable feature measurements.

To illustrate the resulting behavior we created two similar graphs shown in figure 4.12. The two sparse graphs are supposed to represent the same object but have – due to noise or occlusion – a different structure. In particular, the node with the label 11 in the left graph does not appear in the right graph where it has been replaced by node 12. Some affected edges were changed too. In the context of object recognition, the desired matching should preserve as much as possible the relative position of the nodes resulting in the following matching: $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) \mapsto (1, 11, 2, 3, 4, 5, 6, 7, 8, 9, 12, 10)$. The matching shown was obtained by globally optimizing (4.7) and results in the for object recognition undesired matching. It can be seen that this matching – in accordance with the preferred matching of the objective function – maps large edges to each other.

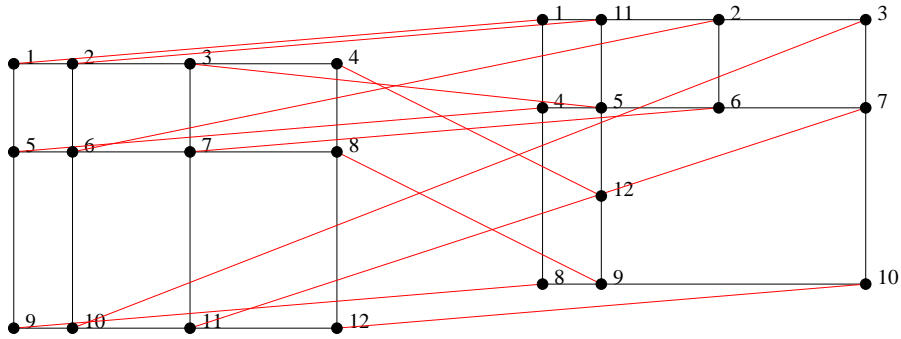


Figure 4.12: The optimal matching for the shown two sparse graphs leads to an undesired matching.

This drawback of the QAP graph matching approach can be tackled by adding

more information to the graphs as has been done in figure 4.13 where the graphs are fully connected. The optimal solution represented by the assignment shown in figure 4.13 coincides with the desired matching.

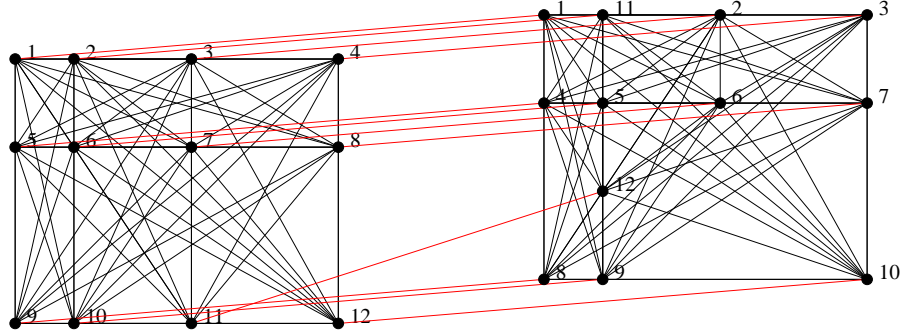


Figure 4.13: The optimal matching for the shown two fully connected graphs leads to a matching one hopes to achieve.

This example reveals that the QAP graph matching approach is sensitive to large perturbations of the graphs, especially to perturbations that change the order of edges with larger weights. However, this effect can be reduced if it is possible to lower the relative error of the perturbation by adding more information to the created graphs. Another weakness of the QAP graph matching approach is that it is based on the assumption that the matched graphs have the same number of nodes. Below, we discuss approaches to overcome this drawback.

4.8 Towards Subgraph Matching

In this section we outline two efforts to adapt the QAP graph matching approach to make it appropriate also for subgraph matching. Note that we have postponed this attempts in favor of the subgraph matching approach suggested in the next chapter, but we discuss the ideas and the progress we have made towards a weighted subgraph matching.

4.8.1 A Simple Extension Approach

The first subgraph matching approach we propose simply fills up the smaller graph with *virtual nodes* such that the QAP graph matching approach for equally sized graphs can be applied. Assuming that the adjacency matrices $A_G \in \mathbb{R}^{n \times n}$ and $A_H \in \mathbb{R}^{m \times m}$ of the two graphs have different sizes $n > m$, the virtual nodes are introduced by extending the smaller adjacency matrix with zeros until the size of the larger matrix is reached:

$$A_H \in \mathbb{R}^{m \times m} \rightarrow \hat{A}_H = \begin{pmatrix} A_H & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n} \quad (4.33)$$

Using exactly the same approach as for graph matching with graphs of the same size (cf. (4.6)), we have to minimize the following objective function:

$$\begin{aligned} f_1(X) &= \|X^\top A_G X - \hat{A}_H\|^2 \\ &= C_G + \hat{C}_H - 2\text{Tr}[A_G^\top X \hat{A}_H X^\top] \end{aligned} \quad (4.34)$$

Here C_G and $\hat{C}_H = \text{Tr}[\hat{A}_H \hat{A}_H^\top]$ are constant values. Our subgraph matching approach corresponds then to the minimization of the objective function (4.34) over the set of permutation matrices $X \in \Pi$:

$$\min_{X \in \Pi} f_1(X) \quad (4.35)$$

Figure 4.14 depicts a small subgraph matching problem: The left triangle should be matched to the right graph containing one more node. Note that the desired matching represents a subgraph isomorphism. The matching obtained by the global minimization of (4.35) leads to the depicted undesired matching.

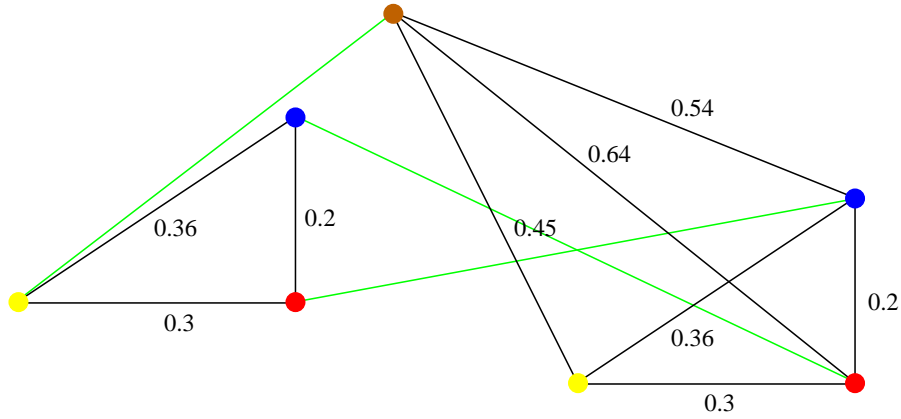


Figure 4.14: Undesired matching found for the shown graphs by the subgraph matching approach which introduces virtual nodes. The matching is obtained by the minimization of (4.35).

One can see that the reason which leads to the undesired matchings discussed in section 4.7.2 is also responsible for the malfunction of the approach applied to the small subgraph matching problem. The undesired matching confirms the fact that the minimization of (4.34) prefers matchings where the most weighted edges are in correspondence to each other. Unfortunately this means that it is likely that the method to introduce virtual nodes results in a non-desired matching. In the following section we discuss an adapted objective function which is able to cope with subgraph matching.

4.8.2 A Promising Subgraph Matching Approach

Using matching matrices $X \in \mathcal{M}^{n \times m}$ (cf. section 2.2.2) instead of permutation matrices, an objective function can be defined which models the weighted graph matching for graphs with different sizes in a more appropriate way. Assuming

that the adjacency matrices $A_G \in \mathbb{R}^{n \times n}$ and $A_H \in \mathbb{R}^{m \times m}$ of the two graphs have different sizes $n > m$, we define the following objective function:

$$\begin{aligned} f_2(X) &= \|X^\top A_G X - A_H\|^2 \\ &= \text{Tr}[X^\top A_G X X^\top A_G^\top X] + \text{Tr}[A_H^\top A_H] - 2\text{Tr}[A_H X^\top A_G^\top X] \\ &= \text{Tr}[X^\top A_G X X^\top A_G^\top X] + C_H - 2\text{Tr}[A_G^\top X A_H X^\top] \end{aligned} \quad (4.36)$$

The minimization of this objective function over all matching matrices $X \in \mathcal{M}$ represents our second subgraph matching approach:

$$\min_{X \in \mathcal{M}} f_2(X) \quad (4.37)$$

As shown in figure 4.15 the optimization approach (4.37) now leads to the desired matching for the small subgraph matching problem. The subgraph isomorphism is found and note that in such a case the objective function (4.36) attains zero as minimum value.

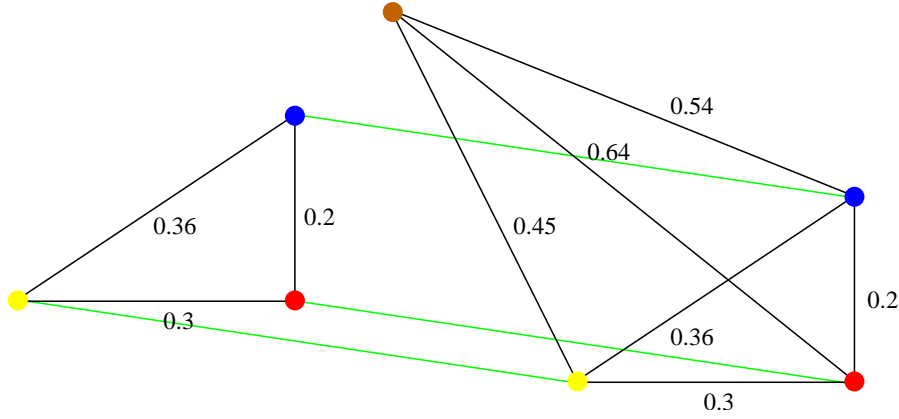


Figure 4.15: The desired matching is obtained by the minimizing (4.36) over all possible matching matrices $X \in \mathcal{M}^{4 \times 3}$.

The only but important difference between the approaches (4.37) and (4.35) is that X in (4.37) is now a matching matrix instead of a permutation matrix. Therefore, the first term in (4.36) is no longer constant. Recognizing that the matching matrix removes $n - m$ rows and columns in A_G

$$A_G \in \mathbb{R}^{n \times n} \rightarrow X^\top A_G X \in \mathbb{R}^{m \times m},$$

the intuitive idea behind the objective function (4.36) becomes apparent. This objective function removes $n - m$ vertices from the larger graph and the remaining edge-weights of the two same sized graphs ($|V_{X^\top A_G X}| = |V_{A_H}|$), are compared.

Note that the invariance against rotation and translation of the graphs is preserved but the property of scale invariance is lost as a scaling of one of the graphs changes the ratio of the first term to the third term in (4.36).

4.8.3 Comparison of the two Approaches

The objective functions (4.34) and (4.36) of the two proposed subgraph matching approaches are compared in figure 4.16. We plotted $f_1(\hat{X})$ and $f_2(X)$ for the small subgraph isomorphism example shown in figures 4.14 and 4.15. We can compare the two functions because for every matching matrix $X \in \mathcal{M}^{4 \times 3}$ an appropriate permutation matrix $\hat{X} = (XS) \in \Pi^{4 \times 4}$ can be found which basically corresponds to the matching matrix X . The matching matrices and the appropriate permutation matrices on the y-axis are sorted according to the objective values of $f_2(X)$.

Comparison of two Objective Functions for Subgraph Matching

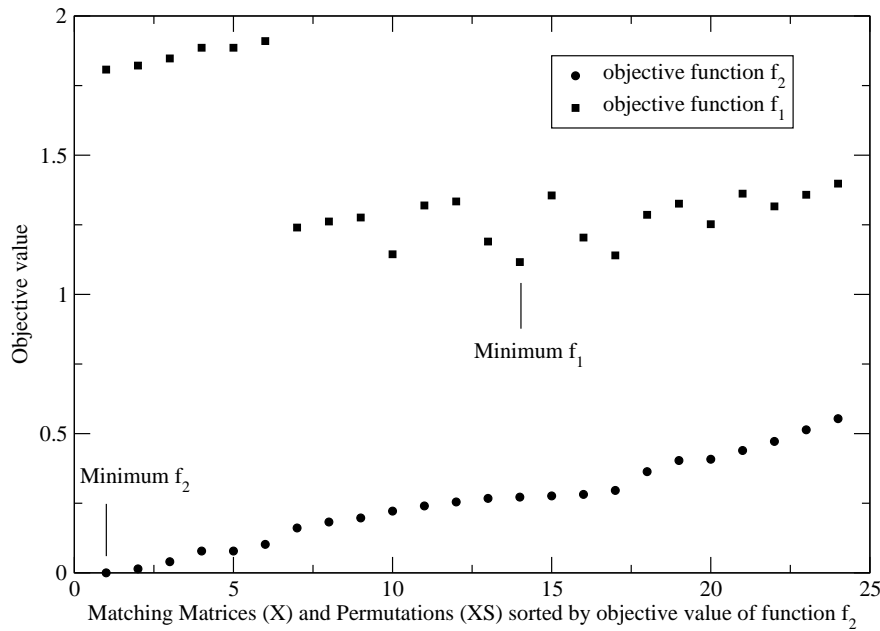


Figure 4.16: Comparison of the two objective functions proposed for weighted subgraph matching. The minimum of f_2 (4.36) corresponds to a desired matching while the minimum of f_1 (4.34) leads to an undesired matching.

The function f_2 (4.36) achieved its minimum at the desired assignment while the minimum of f_1 (4.34) leads to a complete undesired matching.

Comparing the two objective functions reveals that the term $\text{Tr}[X^\top A_G X X^\top A_G^\top X]$ in (4.36) is the important adjustment that leads to the desired optimal solution of the subgraph matching approach (4.37). Note that the QAP term is not influenced by the transition from permutation matrices $\hat{X} = (XS) \in \Pi$ to matching matrices $X \in \mathcal{M}$ and $-2\text{Tr}[A_G^\top \hat{X} \hat{A}_H \hat{X}^\top] = -2\text{Tr}[A_G^\top X A_H X^\top]$ is valid.

If a subgraph isomorphism is present the subgraph matching approach (4.36) attains zero – the smallest possible value – when the isomorphism is found.

For small perturbations of the graphs it is likely that the approach leads to a desired matching too. This supports the assumption that (4.37) represents a reasonable approach for weighted subgraph matching, while the small example in figure 4.14 has shown that the approach to introduce virtual nodes (4.35) is not an appropriate subgraph matching approach.

Note that the combinatorial subgraph matching approach (4.37) is NP-hard as it includes the NP-hard QAP graph matching approach as special case ($n = m$). In the next section we make a proposal how this combinatorial subgraph matching problem can eventually be approximated by a convex relaxation.

4.8.4 Relaxation Attempt

In section 4.8.2 we proposed the following promising integer minimization problem for the weighted subgraph matching for graphs with sizes $n > m$:

$$\min_{X \in \mathcal{M}} \text{Tr}[X^\top A_G X X^\top A_G^\top X] + C_H - 2\text{Tr}[A_H X A_G^\top X^\top] \quad (4.38)$$

$A_G \in \mathbb{R}^{n \times n}$ and $A_H \in \mathbb{R}^{m \times m}$ are the adjacency matrices of the graphs and $X \in \mathcal{M}^{n \times m}$ denotes a matching matrix.

We propose a relaxation approach to the integer minimization (4.38). Note that we have postponed this approach to future research and investigated in favor to this the convex subgraph matching approach we propose in chapter 5. Therefore we sketch only the idea to obtain a convex relaxation of (4.38) along with some methods that could be useful to solve the proposed relaxation. Our idea is based on the observation that the first term $\text{Tr}[X^\top A_G X X^\top A_G^\top X]$ in (4.38) is convex and that

$$-2\text{Tr}[A_G^\top \hat{X} \hat{A}_H \hat{X}^\top] = -2\text{Tr}[A_G^\top X A_H X^\top] \quad (4.39)$$

is valid. Recall that \hat{A}_H is an extended adjacency matrix (see (4.33)) and $\hat{X} = (XS) \in \Pi^{n \times n}$ is a permutation matrix which consists of the matching matrix $X \in \mathcal{M}^{n \times m}$ that is augmented by an appropriate slack matrix $S \in \mathcal{M}^{n \times (n-m)}$. The equality (4.39) allows the application of the convex quadratic relaxation of Anstreicher and Brixius [21] to the QAP term $-2\text{Tr}[A_G^\top \hat{X} \hat{A}_H \hat{X}^\top]$ which results in the following convex relaxation (see section 4.2.4):

$$\begin{aligned} \min \quad & \text{vec}(\hat{X})^\top \hat{Q} \text{vec}(\hat{X}) \\ \text{s.t.} \quad & \hat{X}e = \hat{X}^\top e = e \\ & \hat{X} \geq 0 \end{aligned} \quad (4.40)$$

Maybe the “slack” elements of $\text{vec}[(XS)]$ can be removed from the problem along with the appropriate parts of the matrix \hat{Q} in this convex program without affecting the convexity of the optimization problem.

Recognizing that $\text{Tr}[X^\top A_G X X^\top A_G^\top X]$ is already convex we add this term (and the constant C_H) to the objective function of (4.40) without changing the convexity of the optimization problem. As for matching matrices only the column

sums are fixed to one we expect to obtain a relaxation of the following form:

$$\begin{aligned}
\min \quad & \text{Tr}[X^\top A_G X X^\top A_G^\top X] + C_H + \text{vec}(X)^\top Q \text{vec}(X) \\
\text{s.t.} \quad & X^\top e = e \\
& X^\top X = I \\
& X \geq 0
\end{aligned} \tag{4.41}$$

The resulting optimization problem is hopefully a convex and tight relaxation of (4.38). Regardless of the convexity properties of (4.41) a local optimum of (4.41) can be computed by minimizing the objective function of (4.41) over the *Stiefel manifold* [135] along with the additional constraint that all row sums of X are one. Below we sketch methods which might become useful to solve optimization problems with such constraints.

Minimization on the Stiefel Manifold

Following mainly the paper of Edelman [38] we sketch an easy to implement gradient descent algorithm to minimize a function $F(X)$ over the Stiefel manifold $X \in \mathcal{Q}$. A local minimum can be found by starting at a feasible $X \in \mathcal{Q}$ and computing the descent gradient direction H of the function $F(X)$ in this point. Then while staying on the Stiefel manifold the minimum of $F(X)$ in that direction is searched and used as new start point. This steps are repeated until the algorithm converges to the local minimum.

According to [38] the descent gradient direction – using the so called *canonical metric* – can be computed by

$$H = -\nabla F(X) = -(F_X - X F_X^\top X)$$

where the elements of F_X are the partial derivatives of F with respect to the elements of X :

$$(F_X)_{ij} = \frac{\partial F}{\partial X_{ij}}$$

The geodesic on the Stiefel manifold starting from X_0 in the gradient direction H is given by the curve

$$X(t) = X_0 M(t) + Q N(t)$$

were

$$QR = (I - X X^\top) H \tag{4.42}$$

is the QR-decomposition and $M(t)$ and $N(t)$ can be determined by the matrix exponential

$$\begin{pmatrix} M(t) \\ N(t) \end{pmatrix} = \exp \left[t \begin{pmatrix} A & -R^\top \\ R^\top & 0 \end{pmatrix} \right] \begin{pmatrix} I \\ 0 \end{pmatrix}$$

with $A = Y^\top H$ and R from the QR-decomposition (4.42). The matrix exponential can be calculated for example by an algorithm suggested in [64]. With this the minimum of $F(X(t))$ in the direction $H = -\nabla F(X)$ can be searched and used as starting point for the next iteration of this steps.

In [38] one can find a good and detailed description for the Newton and conjugate gradient algorithm on the Stiefel manifold.

4.8.5 Projection Approach

The aim of this section is to present a projection method we have found for relaxed matching matrices which allows the substitution of $X \in \mathcal{Q}^{n,m} \cap \bar{\mathcal{E}}$ by expressing X in terms of $y \in \mathcal{Q}^{n-1,m}$. Here \mathcal{Q} denotes the Stiefel manifold ($X^\top X = I$) and $\bar{\mathcal{E}}$ denotes the set of matrices with column sum one ($X^\top e = e$). The discussed approach is inspired by the projection approach proposed by Hadley et al. [59] which takes into account the doubly stochastic constraint $X \in \mathcal{E}$ in addition to the orthogonality constraint $X \in \mathcal{O}$ for relaxed permutation matrices (see also section 4.2.3). Setting

$$X = \tilde{X} + V_n z \quad (4.43)$$

where

$$\tilde{X} = \frac{1}{n} E_{nm} = \frac{1}{n} e_n e_m^\top$$

we want X to be within the Stiefel manifold:

$$X^\top X = \frac{1}{n} E_{mm} + z^\top z \stackrel{!}{=} I_{n-1,n-1}$$

We observe, as $z^\top z = I_{n-1,n-1} - \frac{1}{n} E_{mm}$, that z itself is not within the Stiefel-manifold and the question arises if it can be expressed as a function $z(y)$ which depends on a variable y that lies in the Stiefel-manifold. Indeed it turns out that this can be achieved by the following expression:

$$z(y) = y(I_{mm} - cE_{mm})$$

Solving the quadratic equation $z^\top z = I - 2cE_{mm} + c^2 m E_{mm} \stackrel{!}{=} I - \frac{1}{n} E_{mm}$ shows, that with a c set to one of the following two positive values ($n > m$)

$$c_{1,2} = \frac{1}{m} \pm \sqrt{\frac{1}{m^2} - \frac{1}{mn}}$$

z can really be expressed as a function of a variable y that lies in the Stiefel-manifold. Therefore

$$X(y) = \frac{1}{n} E_{nm} + V_n y(I_{mm} - cE_{mm})$$

allows to express $X \in \mathcal{Q}^{n,m} \cap \bar{\mathcal{E}}$ in terms of $y \in \mathcal{Q}^{n-1,m}$. Note that due to the ambiguity of c one has to cope with the two possible substitutions of X .

Chapter 5

Subgraph Matching

In this chapter, we present a new convex programming approach to the problem of subgraph matching. The aim is to match object views, represented by graphs, against larger graphs which represent scenes. Starting from a linear programming formulation for computing the optimal bipartite matching between the nodes of the two graphs, we extend the linear objective function in order to take into account the relational constraints given by both graphs. The resulting combinatorial optimization problem is approximately solved by a semidefinite program. Results of numerous experiments evaluating the approach are presented.

5.1 Problem Statement

The success of the convex relaxation approach for graph matching as investigated in the previous chapter motivates to move on and to formulate an equivalent approach to *subgraph* matching problems. Figure 5.1 depicts an example of a subgraph matching problem: the smaller graph on the left side has to be matched against the larger graph on the right side. We propose a subgraph

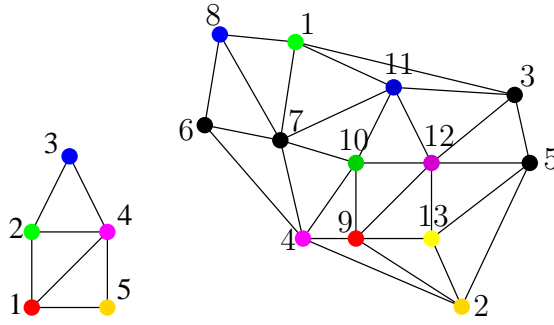


Figure 5.1: A subgraph matching problem: The model graph with $K = 5$ (left) has to be matched against the scene graph with $L = 13$ nodes (right). Similar nodes in the model graph and in the scene graph are indicated by similar colors.

matching approach which incorporates the similarity between the nodes of the two involved graphs *along with the underlying structure* of the graphs. To this

end, we assume that the similarities between all pairs of nodes of the two graphs can be calculated. In figure 5.1 this similarity is displayed by the colors of the nodes. Moreover, the relational structure of the object and the scene is assumed to be represented by simple non-weighted graphs.

We point out again that our focus in this thesis is on the development of good approximation algorithms for graph matching on the basis of convex programming techniques. The decisive advantages of convex programming techniques are that no additional tuning parameters, which have to be optimized, are introduced and that convex programs can be solved reliably by established standard methods like interior point algorithms. As already mentioned in section 1.2.4, we do not consider in-depth issues related to image preprocessing and assume the model and scene graphs to be given.

Starting from a bipartite graph matching (section 5.2), we extend the linear objective criterion with a quadratic term favoring bipartite matchings which take into account the relational structure of both the model graph and the scene graph. This results in a quadratic integer program which naturally is NP-hard (section 5.3). In section 5.4, a semidefinite relaxation of this combinatorial problem is developed. An illustrative example is discussed in section 5.5 followed by various numerical experiments in section 5.6 and 5.7. In section 5.8 some real world subgraph matching problems are evaluated. Several interesting aspects of the subgraph matching approach are discussed in section 5.9.

5.2 Notation and Bipartite Matching

In the following, we briefly recapitulate the basic notation used in this chapter and provide the starting point of our approach, which consists of the bipartite matching between two sets of nodes which originate from the object and scene graph.

5.2.1 Matching in Bipartite Graphs

We consider undirected graphs $G = (V, E)$ with nodes $V = \{1, \dots, n\}$ and edges $E \subset V \times V$. We denote the model graph with G_K and the scene graph with G_L . The corresponding sets V_K and V_L contain $K = |V_K|$ and $L = |V_L|$ nodes respectively. We assume the scene graph to be larger than the object graph ($L \geq K$). Furthermore, we assume a similarity function $w(i, j)$ to be given which measures the similarity of each pair of nodes $i \in V_K$ and $j \in V_L$.

If we ignore the structure in both the model graph and the scene graph, then an optimal assignment of the K nodes of the model graph to the scene graph can be found easily as a bipartite matching between the nodes V_K and V_L (cf. section 2.2.1).

The bipartite matching is to find a matching in the bipartite graph $G_{bipartite} = (V_K \cup V_L, E)$, which has edges $(i, j) \in E$ defined for all pairs $i \in V_K, j \in V_L$ with the corresponding edge weights $w(i, j)$. This means that $G_{bipartite}$ consists of $K + L$ nodes and has KL undirected edges linking every node in V_K with every node in V_L . Note that no edge of $G_{bipartite}$ links together nodes within the sets

V_K and V_L , respectively. The optimal bipartite matching can be computed by a linear optimization problem which is described in the following section.

5.2.2 Linear Problem Formulation of the Bipartite Matching

In this section, we sketch a linear programming approach to compute a bipartite matching with minimal cost between two sets of nodes. To this end, the matching is described by a 0/1-indicator vector x . Let $x \in \{0, 1\}^{KL}$ be the indicator vector with elements x_{ji} for all pairs $i \in V_K, j \in V_L$. The element x_{ji} represents the edge between the node $i \in V_K$ and $j \in V_L$. An edge (i, j) is present if $x_{ji} = 1$, and $x_{ji} = 0$ indicates that an edge is not present. We assume that the components of x are arranged as described in section 2.2.2:

$$x = (x_{11}, \dots, x_{L1}, x_{12}, \dots, x_{L2}, \dots, x_{1K}, \dots, x_{LK})^\top. \quad (5.1)$$

Thus x starts with L edges connecting the first node of V_K with all nodes in V_L , followed by L edges connecting the second node in V_K with all nodes in V_L , etc. We denote, with a slight abuse of notation, the corresponding weight vector $(w(1, 1), \dots, w(L, K))^\top$ again with w . The optimal bipartite matching is then obtained by solving the following integer program

$$\begin{aligned} \min_x \quad & w^\top x \\ \text{s.t.} \quad & A_K x = e_K, \quad A_L x \leq e_L \\ & x \in \{0, 1\}^{KL} \end{aligned} \quad (5.2)$$

where the matching constraints are defined by the constraint matrices A_L and A_K . Conforming to section 3.5, the matrix $A = (A_K^\top, A_L^\top)^\top$ composed of A_L and A_K , is the incidence matrix of the bipartite graph $G_{bipartite}$. Hence, A is a totally unimodular matrix [108] and, as a consequence, (5.2) can be solved by a linear program where $x \in \mathbb{R}^{KL}$ and its solution is guaranteed to result in the globally optimal integer solution $x \in \{0, 1\}^{KL}$. Therefore we refer to the integer program (5.2) often as linear program. As mentioned above, however, this favorable situation has been achieved by ignoring the relational structures of the object representation (model graph) and the image (scene graph). Applying the linear approach (5.2) to the subgraph matching example shown in figure 5.1, each node of the object graph is assigned to the best fitting partner. However, without taking the structure into account, this leads to the undesired matching shown in figure 5.2.

In order to include the relational constraints of the graphs we extend the linear approach (5.2) in the next section. In section 5.4 we propose a corresponding convex programming relaxation which is also favorable from the computational point of view.

5.3 Combinatorial Subgraph Matching Approach

In this section, we suggest a formulation of the subgraph matching problem as a quadratic integer program which is based on a regularization of the previously

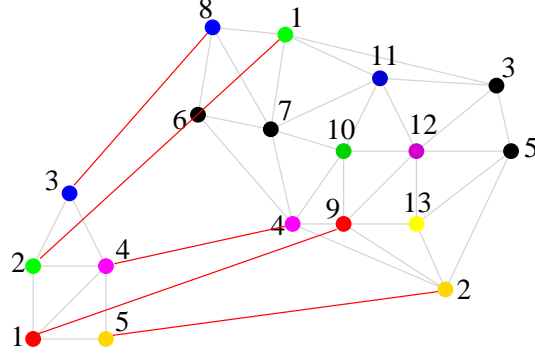


Figure 5.2: Ignoring the structure of the graphs the solution of the linear optimization problem (5.2) leads to the shown undesired matching.

discussed bipartite matching. The quadratic program is discussed in detail in section 5.3.1. After a short discussion of the regularization parameter in section 5.3.2, the approach is classified in section 5.3.4. In order to find a good approximation of this combinatorial subgraph matching problem, it will be relaxed to a convex optimization problem in section 5.4.

5.3.1 Quadratic Integer Program

To incorporate the relational structure of both the model graph and the scene graph, we extend the linear objective function in (5.2) with a quadratic term. To control the influence of these additional costs the non-negative parameter $\alpha \in \mathbb{R}^+$ is introduced. Formally the quadratic integer program then reads:

$$\begin{aligned} \min_x \quad & w^\top x + \alpha x^\top Q x \\ \text{s.t.} \quad & A_K x = e_K, \quad A_L x \leq e_L \\ & x \in \{0, 1\}^{KL} \end{aligned} \tag{5.3}$$

As before, the matching constraints are defined by the linear constraints. The matrix $Q \in \mathbb{R}^{KL \times KL}$ in the quadratic term of (5.3) to be specified below involves the symmetric 0/1-adjacency matrices N_K, N_L of the model graph and the scene graph, respectively, which encode the neighborhood structure in these two graphs. For example, the adjacency matrix N_K for the house-like model graph from figure 5.1 is shown in figure 5.3.

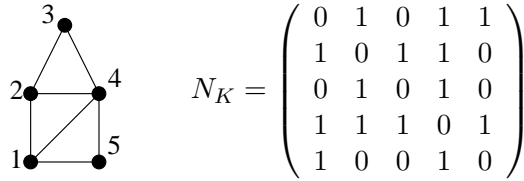


Figure 5.3: Example object graph and its adjacency matrix N_K .

Furthermore we define the *Complementary Adjacency Matrices*.

Definition 1. Complementary Adjacency Matrices

$$\begin{aligned}\bar{N}_L &= E_{LL} - N_L - I_L \\ \bar{N}_K &= E_{KK} - N_K - I_K\end{aligned}$$

These matrices can be interpreted as indicator matrices for *non-adjacent* nodes. They have the element $(\bar{N})_{ij} = 1$ if the corresponding nodes i and j are not directly connected in the graph. With this notation and referring to the order of the set of edges defined in (5.1), the symmetric *Relational Structure Matrix* Q in (5.3) incorporating the relational structure is defined in the following.

Definition 2. Relational Structure Matrix

$$Q = N_K \otimes \bar{N}_L + \bar{N}_K \otimes N_L \quad (5.4)$$

We explain in detail the two terms on the right hand side of (5.4) which are used to construct the matrix Q :

- The first term in the quadratic expression $x^\top Q x$ can be written as:

$$x^\top (N_K \otimes \bar{N}_L) x = \sum_{ar} \sum_{bs}^{KL} (N_K)_{ab} (\bar{N}_L)_{rs} x_{ar} x_{bs} \quad (5.5)$$

The interpretation of this term is that if two nodes a and b in the model graph are neighbors, $(N_K)_{ab} = 1$, then a good assignment (no costs) involves corresponding nodes r and s in the scene graph which are neighbors, too: $(\bar{N}_L)_{rs} = 0$. For such a configuration no cost is added in (5.5). Otherwise if the corresponding nodes r and s are no neighbors in the scene graph, $(\bar{N}_L)_{rs} = 1$, then a cost of 1 is added. This two configurations are visualized in figure 5.4.

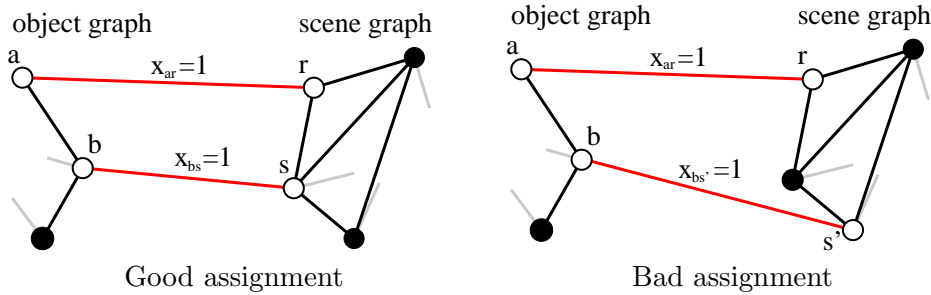


Figure 5.4: **Left:** Adjacent nodes a and b in the model graph are assigned to adjacent nodes r and s in the scene graph. **Right.** Adjacent model nodes a and b are no longer adjacent in the scene graph after the assignment. The left assignment leads to no additional costs while the right undesired assignment adds 1 to the cost term (5.5).

- Analogously, the second term in $x^T Q x$ gives:

$$x^T (\bar{N}_K \otimes N_L) x = \sum_{ar} \sum_{bs}^{KL} (\bar{N}_K)_{ab} (N_L)_{rs} x_{ar} x_{bs} \quad (5.6)$$

This term penalizes assignments where pairs of nodes in the object graph become neighbors in the scene graph which were not adjacent before. Figure 5.5 illustrates this situation in detail.

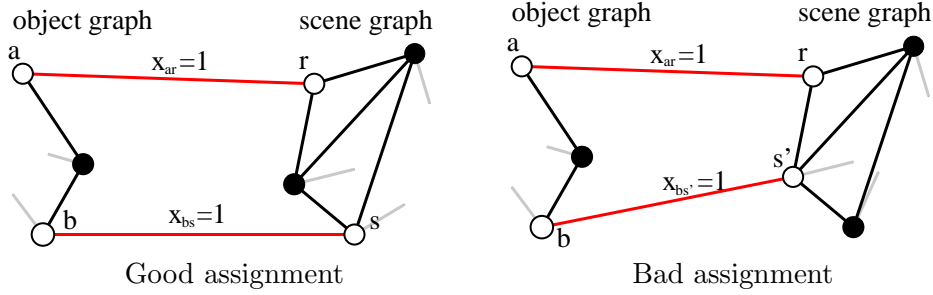


Figure 5.5: Left: Nodes a and b which are not adjacent in the object graph are assigned to nodes which are also not adjacent in the scene graph. **Right:** A pair of nodes a and b become neighbors r and s' after assignment. The left assignment is associated with no additional costs in (5.6). The undesired assignment on the right side adds 1 to these costs.

Note that due to the symmetry of the quadratic cost term $x^T Q x$, every difference in the compared structure of the two graphs is penalized with a cost of 2. Figure 5.6 shows the matching which is obtained for the subgraph matching example from figure 5.1 using the combinatorial subgraph matching approach (5.3). Details about the experiment are described in section 5.6. The important fact is that the additional quadratic term is able to correct the bipartite matching appropriately such that the minimization of (5.3) actually leads to the desired matching.

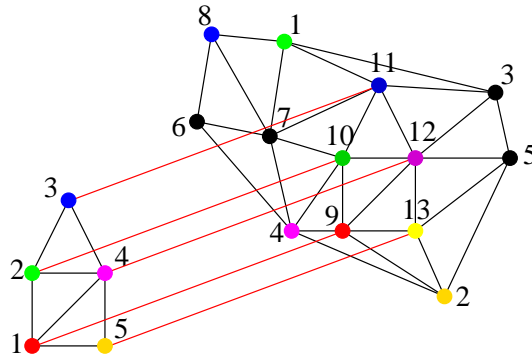


Figure 5.6: The minimization of (5.3) which incorporates the structure of the graphs leads to the desired matching.

The quadratic integer optimization problem (5.3) is a NP-hard problem (cf. section 2.3.2). In order to calculate a good approximate solution, we will derive

a convex relaxation in section 5.4. Before that, we discuss the influence of the regularization parameter α for our subgraph matching approach.

5.3.2 Regularization Parameter

This section concerns the non-negative regularization parameter α which was introduced as scale parameter to the quadratic term in the objective function of (5.3). This parameter adjusts the importance of the quadratic structure term relative to the linear matching cost term. In other words, it controls how much the graph structure is taken into account. If α is increased the structure becomes more important.

Choice of the Parameter

If the elements of the similarity vector w are normalized to values within the range between 0 and 1 experiments showed that setting α to a value between 0.01 and 0.4 is a good choice for the parameter. Of course it should not be too close to zero $\alpha \rightarrow 0$ as this results in the bipartite matching case. If the parameter has a very large value $\alpha \rightarrow \infty$ this means that only the relational structure is considered by our subgraph matching approach. These two extreme cases are discussed in the following.

Bipartite Matching ($\alpha \rightarrow 0$)

In the first case, $\alpha \rightarrow 0$, the quadratic term in the objective function of (5.3) is switched off. It is easy to see that this case results in the bipartite matching problem (5.2) which, as discussed in section 5.2.2, ignores any knowledge of the underlying relational structure of the graphs.

Subgraph Isomorphism ($\alpha \rightarrow \infty$)

The second case, $\alpha \rightarrow \infty$, makes the first term in the objective function of (5.3) negligible, and only the quadratic term remains. Therefore this case can be studied by setting all similarities in w to zero¹ and α to an arbitrary value, e.g. $\alpha = 1$:

$$\begin{aligned} \min_x \quad & x^\top Q x \\ \text{s.t.} \quad & A_K x = e_K, \quad A_L x \leq e_L \\ & x \in \{0, 1\}^{KL} \end{aligned} \tag{5.7}$$

In the case of a subgraph isomorphism the combinatorial solution of (5.7) leads to the lowest possible objective value zero. Note that conversely an objective value of zero proves that a subgraph isomorphism is present in a given problem instance. We can see that the minimization of (5.7) represents the search for a matching which has the smallest possible deviation from a subgraph isomorphism. The question whether the convex relaxation of (5.7) leads to a bound

¹Setting all similarities to equal values has the same effect, as this only adds a constant to the objective function.

which can be used to decide if a subgraph isomorphism does not occur in a given problem instance is investigated in section 5.9.5.

5.3.3 Hidden Parameters

There are hidden parameters that do not belong directly to our subgraph matching approach, but which influence the results. In particular, one can observe that the edge density of the graph structures changes the capability of our approach to obtain good subgraph matchings. In this thesis we set the edge probabilities to reasonable values during the creation process without investigating the effect of changing problem creation parameters.

Another example is the accuracy of the similarity measure between the nodes of the two graphs which is crucial for obtaining the desired mapping. We chose similarity values here that seem reasonable to us to occur in subgraph matching problems.

These parameters are related to the process of creating or obtaining graphs and important for the question how one can build object and scene graphs in a reliable way. However, as already mentioned in section 1.2.4 dealing with this question is beyond the scope of this work.

5.3.4 Classification of the Approach

The combinatorial subgraph matching approach (5.3) with Q defined in (5.4) belongs to the category of *error correcting graph matching*. These approaches intend to minimize graph edit distances (see section 1.6.2). As discussed previously, the quadratic cost term in (5.3) penalizes every difference between the structure of the object graph and the mapped subgraph in the scene with costs of 2. Therefore, the objective function in (5.3) is a measurement of the structural differences and can be interpreted as edit cost for deleting and creating appropriate edges in the object graph needed to obtain a graph which is isomorphic to the mapped subgraph in the scene. The edit costs can be adjusted by the regularization parameter α . If a subgraph isomorphism is attained the quadratic term is zero. Note that this approach is only reasonable for non-fully connected scene graphs as a fully connected scene graph contains no usable structure information. In the latter case, the quadratic term $x^\top Q x$ is constant for every matching which reduces the quadratic approach (5.3) to the linear bipartite matching (5.2). In real scenes, however, locally connected graphs are typically used to represent spatial context in images.

5.4 Semidefinite Program Formulation

In contrast to the linear bipartite matching problem (5.2), the computation of the global optimum of the quadratic optimization problem (5.3), which incorporates the object and scene structure, is intrinsically difficult. Therefore, we derive a tractable convex relaxation of this NP-hard problem in order to compute a “good” local minimum. The combinatorial subgraph matching approach

(5.3) will be relaxed to a (convex) semidefinite program (SDP) which has the following standard form:

$$\begin{aligned}
 \min \quad & \text{Tr} [\tilde{Q}X] \\
 \text{s.t.} \quad & \text{Tr}[A_1X] = c_1 \\
 & \text{Tr}[A_2X] = c_2 \\
 & \vdots \\
 & \text{Tr}[A_mX] = c_m \\
 & X \succeq 0
 \end{aligned} \tag{5.8}$$

The last constraint in (5.8) says that X has to be positive semidefinite. We wish to emphasize once more that this convex optimization problem can be solved with standard methods like interior point algorithms. Below, we describe step by step how we derive such a semidefinite program from (5.3) while in section 5.4.1, we derive an appropriate SDP objective function. We show in section 5.4.2 how the bipartite matching constraints can be incorporated into the SDP (5.8). After discussing SDP inequality constraints in section 5.4.3 we propose a post-processing step in section 5.4.4 to obtain a close integer solution from the non-integer solution of the convex relaxation (5.8).

5.4.1 Objective Function

In order to obtain an appropriate SDP relaxation for the combinatorial subgraph matching problem, we start with reformulating the objective function of (5.3) into a homogeneous quadratic form

$$f(x) = w^\top x + \alpha x^\top Q x = \begin{pmatrix} 1 & x^\top \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2}w^\top \\ \frac{1}{2}w & \alpha Q \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}, \tag{5.9}$$

which can be stated directly in the desired trace formulation

$$f(x) = \text{Tr} \left[\begin{pmatrix} 0 & \frac{1}{2}w^\top \\ \frac{1}{2}w & \alpha Q \end{pmatrix} \begin{pmatrix} 1 & x^\top \\ x & xx^\top \end{pmatrix} \right], \tag{5.10}$$

where we have used the cyclic commutativity of the trace. To simplify this expression we denote with $\tilde{Q} \in \mathbb{R}^{(KL+1) \times (KL+1)}$ and $X \in \mathbb{R}^{(KL+1) \times (KL+1)}$ the following symmetric matrices:

$$\tilde{Q} = \begin{pmatrix} 0 & \frac{1}{2}w^\top \\ \frac{1}{2}w & \alpha Q \end{pmatrix}, \quad X = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 & x^\top \end{pmatrix} = \begin{pmatrix} 1 & x^\top \\ x & xx^\top \end{pmatrix} \tag{5.11}$$

Besides being symmetric, the matrix X is positive semidefinite and has rank 1. We relax the objective function by dropping the rank 1 condition of X which makes the set of feasible matrices convex [146]. This lifts the original problem (5.3) defined in a vector space with dimension KL into the space of symmetric, positive semidefinite matrices with the dimension $(KL+1) \times (KL+1)$. Using \tilde{Q} and X we have

$$f(x) = \text{Tr} [\tilde{Q}X]$$

which represents the appropriate objective function for the semidefinite relaxation (5.8). In the following section, we are concerned with the task to obtain suitable SDP constraints which represent the constraints of the subgraph matching approach (5.3).

5.4.2 Equality Constraints

We wish to incorporate several constraints into the SDP relaxation by specifying appropriate constraint matrices $A_i \in \mathbb{R}^{(KL+1) \times (KL+1)}$. These SDP constraints will have the form:

$$\text{Tr}[A_i X] = c_i \quad \text{for } i = 1, \dots, m$$

In particular, we consider four types of constraints which correspond to the homogeneous formulation of the problem, the 0/1-integer constraints, and the bipartite matching constraints, respectively. We next discuss in detail how these constraints are incorporated into the SDP formulation:

- The first constraint we take into account results from the homogenization (5.9). To restrict the element $X_{11} = 1$ in the matrix X , we introduce a constraint matrix $^{\text{one}}A$ whose elements can be expressed as

$$^{\text{one}}A_{kl} = \delta_{k1}\delta_{l1} \quad \text{for } k, l = 1, \dots, KL + 1,$$

where we make use of the Kronecker delta. Explicitly, this SDP constraint reads:

$$\text{Tr}[^{\text{one}}AX] = \text{Tr} \left[\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & & 0 \end{pmatrix} X \right] = 1$$

- The second type of constraint we consider is derived from the integer constraints $x_i \in \{0, 1\}, i = 1, \dots, KL$, which can be rewritten as $x_i^2 = x_i, i = 1, \dots, KL$. If we consider the matrix X before it is relaxed (see 5.11) we observe that due to $x_i^2 = x_i$ the 0/1-integer elements on the diagonal of X must be equal to the 0/1-integer elements in the first column and row of X . Therefore the 0/1-integer constraints can be *weakly* enforced in the relaxed problem by requiring the first column and row of X to be equal to its diagonal. To implement these constraints, we introduce KL constraint matrices $^{\text{int}}A^j \in \mathbb{R}^{(KL+1) \times (KL+1)}, j = 2, \dots, KL + 1$. We define these constraint matrices to have a 2 at the appropriate diagonal element and -1 at the corresponding elements in the first column and the first row. All other elements are zero. Using the Kronecker delta the elements of the j -th constraint matrix $^{\text{int}}A^j$ are:

$$^{\text{int}}A_{kl}^j = 2\delta_{kj}\delta_{lj} - \delta_{kj}\delta_{l1} - \delta_{lj}\delta_{k1} \quad \text{for } k, l = 1, \dots, KL + 1$$

As a result, we have the following SDP constraints:

$$\text{Tr}[\text{int} A^j X] = 0 \quad \text{for } j = 2 \dots, KL + 1$$

For example, for $j = 2$, this constraint reads:

$$\text{Tr}[\text{int} A^2 X] = \text{Tr} \left[\begin{pmatrix} 0 & -1 & 0 & \cdots & 0 \\ -1 & 2 & 0 & & 0 \\ 0 & 0 & 0 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & & 0 \end{pmatrix} X \right] = 0$$

- The third type of constraint we take into account are the equality constraints $\sum_{j=1}^L x_{ij} = 1, i = 1, \dots, K$, which are part of the bipartite matching constraints in (5.3). They represent the constraint that each node of the smaller graph is mapped to exactly one node of the scene graph. We define K constraint matrices $\text{sum} A^j \in \mathbb{R}^{(KL+1) \times (KL+1)}, j = 1, \dots, K$ which ensure (taking the order of the diagonal elements into account) that the sum of the appropriate portion of the diagonal elements of X is 1. We exploited here the fact that $x_i = x_i^2$ holds true for 0/1-variables. The matrix elements for the j -th constraint matrix $\text{sum} A^j$ can be expressed as follows:

$$\text{sum} A_{kl}^j = \sum_{i=(j-1)L+1}^{jL+1} \delta_{ik} \delta_{il} \quad \text{for } k, l = 1, \dots, KL + 1$$

The K constraints in SDP form are then:

$$\text{Tr}[\text{sum} A^j X] = 1 \quad \text{for } j = 1, \dots, K$$

Explicitly written, this constraint reads:

$$\text{Tr} \left[\begin{pmatrix} 0 & & & \cdots & & 0 \\ & \ddots & & & & \\ & & 0 & & & \\ & & & 1 & & \\ \vdots & & & & \ddots & \vdots \\ & & & & & 1 & 0 \\ 0 & & & \cdots & & & \ddots & 0 \end{pmatrix} X \right] = 1$$

Note that we consider only equality constraints here, but possible ways to formulate inequality constraints like the bipartite matching inequality constraints ($\sum_{i=1}^K x_{ij} \leq 1, \forall j$) as appropriate SDP constraints are discussed in section 5.4.3.

- The fourth type of constraint is related to the observation that the bipartite matching constraints in (5.2) have a direct impact to certain matrix elements of the sub-matrix $\tilde{X} = xx^\top$ of X . If $x \in \{0, 1\}^{KL}$ represents a bipartite matching then certain elements in \tilde{X} must be zero. Affected elements can be determined by inspecting the following two cost terms which penalize matchings that do not meet the bipartite matching constraints.

$$x^\top (I_K \otimes (E_{LL} - I_L))x = \sum_{ar} \sum_{bs}^{KL} (I_K)_{ab} (E_{LL} - I_L)_{rs} x_{ar} x_{bs} \quad (5.12)$$

$$x^\top ((E_{KK} - I_K) \otimes I_L)x = \sum_{ar} \sum_{bs}^{KL} (E_{KK} - I_K)_{ab} (I_L)_{rs} x_{ar} x_{bs} \quad (5.13)$$

The first of these two terms penalizes non-unique assignments of model nodes to scene nodes. Analogously, the second term penalizes assignments where different nodes of the model graph are mapped to the same node in the scene graph. Thus, in summary, the two terms penalize all assignments which do not lead to a bipartite matching. Figure 5.7 illustrates such configurations in detail.

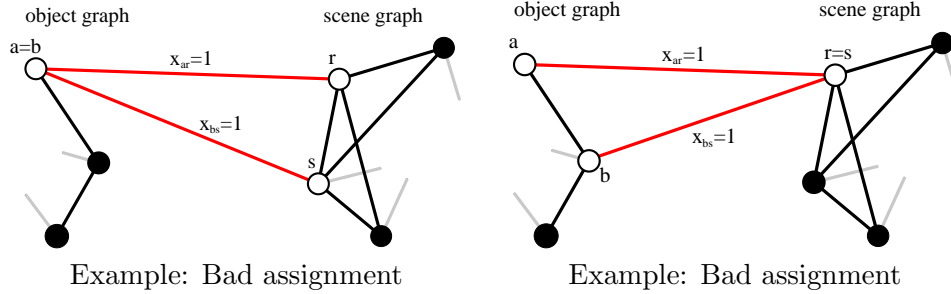


Figure 5.7: Assignments which do not lead to bipartite matchings are penalized by the quadratic terms (5.12) and (5.13).

All integer solutions $\tilde{X} = xx^\top \in \mathbb{R}^{KL \times KL}$, where x represents a bipartite matching, have zero-values at those matrix positions where $I_K \otimes (E_{LL} - I_L)$ and $(E_{KK} - I_K) \otimes I_L$ have non-zero elements. Accordingly, we want to force the corresponding elements in $X \in \mathbb{R}^{(KL+1) \times (KL+1)}$ to be zero. Fortunately, this can be achieved with the constraint matrices $\text{zeros1} A^{ars}$, $\text{zeros2} A^{\hat{s}\hat{a}\hat{b}} \in \mathbb{R}^{(KL+1) \times (KL+1)}$ which are determined by the indices a, r, s and $\hat{s}, \hat{a}, \hat{b}$. They have the following matrix elements

$$\text{zeros1} A_{kl}^{ars} = \delta_{k, (aL+r+1)} \delta_{l, (aL+s+1)} + \delta_{k, (aL+s+1)} \delta_{l, (aL+r+1)} \quad , \quad (5.14)$$

$$\text{zeros2} A_{kl}^{\hat{s}\hat{a}\hat{b}} = \delta_{k, (\hat{s}K+\hat{b}+1)} \delta_{l, (\hat{s}K+\hat{a}+1)} + \delta_{k, (\hat{s}K+\hat{a}+1)} \delta_{l, (\hat{s}K+\hat{b}+1)} \quad , \quad (5.15)$$

where $k, l = 1, \dots, KL + 1$. Each of these matrices has only two non-zero matrix elements at symmetric positions. The indices a, r, s and $\hat{s}, \hat{a}, \hat{b}$ attain all valid combinations of the following triples where $s > r$ and

$\hat{b} > \hat{a}$:

$$(a, r, s) : a = 1, \dots, K; r = 1, \dots, L; s = (r + 1), \dots, L$$

$$(\hat{s}, \hat{a}, \hat{b}) : \hat{s} = 1, \dots, L; \hat{a} = 1, \dots, K; \hat{b} = (\hat{a} + 1), \dots, K$$

This defines the following $(LL - L)K/2 + (KK - K)L/2$ additional constraints

$$\text{Tr}[\text{zeros}^1 A^{ars} X] = 0, \forall (a, r, s)$$

$$\text{Tr}[\text{zeros}^2 A^{\hat{s}\hat{a}\hat{b}} X] = 0, \forall (\hat{s}, \hat{a}, \hat{b})$$

which ensure zero-values at the corresponding matrix positions of X . An explicit example for one of these constraints is shown here:

$$\text{Tr} \left[\begin{pmatrix} 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ & & 0 & 1 & 0 & & \\ \vdots & & 0 & 0 & 0 & & \\ 0 & 0 & 0 & & & & \vdots \\ 0 & 1 & 0 & & \ddots & & \\ 0 & 0 & 0 & & & & \\ \vdots & & & & & & \\ 0 & 0 & & \dots & & & 0 \end{pmatrix} X \right] = 0$$

The name *gangster operator* was introduced in [137] for such constraint operators because it “shoots holes”, i.e. zeros into the matrix X .

Counting all the considered constraints we end up with the fast growing number of constraints:

$$m = 1 + KL + K + \frac{(KK - K)L}{2} + \frac{(LL - L)K}{2} = 1 + K + \frac{K^2L}{2} + \frac{KL^2}{2}$$

So far we considered equality constraints, which can be handled by currently available SDP solvers. However, it is also possible to incorporate inequalities. Therefore, we discuss the issue of reformulating linear inequality constraints into appropriate SDP constraints in the next section.

5.4.3 Inequality Constraints

Inequality constraints can typically be used to further tighten the SDP relaxation. In this section, we summarize possible ways how linear inequality constraints can be stated in an appropriate SDP, similar to (5.8) which additionally incorporates inequality constraints:

$$\text{Tr}[D^i X] \leq b_i \quad \text{for } i = 1, \dots, n$$

In the following, we consider linear inequality constraints

$$Ax \leq b,$$

where $A \in \mathbb{R}^{n \times m}$, $x \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$. Comparing this with the bipartite matching inequality constraints (5.3), we can identify the matrix $A = A_L \in \mathbb{R}^{L \times KL}$ and the vector $b = e_L \in \mathbb{R}^L$. In [68], three different methods – called *lifting* – to obtain SDP constraints from linear inequality constraints involving 0/1-variables are investigated (see also [68, 146, 12.1.2]). The authors showed the following order of tightness

$$\text{Diagonal Lifting} < \text{Squared Lifting} < \text{Lovász Schrijver Lifting},$$

with an increasing number of additional constraints being involved by these approaches. The three ways to incorporate linear inequalities into SDPs are outlined in the following:

Diagonal Lifting

The first and simplest method to incorporate linear inequalities, involving 0/1-variables, is diagonal lifting. Let ${}^i a^\top$ be a row of the original constraint matrix A , then the constraints

$${}^i a^\top x \leq b_i \quad \text{for } i = 1, \dots, n \quad (5.16)$$

can be lifted into the appropriate SDP constraints:

$$\text{Tr}[\text{Diag}({}^i a)X] \leq b_i \quad \text{for } i = 1, \dots, n \quad (5.17)$$

Here, the original inequality $\sum_{j=1}^m {}^i a_j x_j \leq b_i$ is substituted by the quadratic inequality $\sum_{j=1}^m {}^i a_j x_j^2 \leq b_i$ by using the fact that $x_j = x_j^2$ holds true for 0/1-variables.

Squared Lifting

The second lifting approach presumes ${}^i a^\top x \geq 0$ and $b_i \geq 0$ where ${}^i a, x \in \mathbb{R}^m$ and $b_i \in \mathbb{R}$. Then both sides of the n inequalities ${}^i a^\top x \leq b_i, i = 1, \dots, n$ (see (5.16)) are squared which results in the following inequalities:

$${}^i a^\top x x^\top {}^i a \leq b_i^2 \quad \text{for } i = 1, \dots, n \quad (5.18)$$

Substituting xx^\top by $X \in \mathbb{R}^{m \times m}$ leads to the following SDP representation of the inequality constraints:

$$\text{Tr}[{}^i a {}^i a^\top X] \leq b_i^2 \quad \text{for } i = 1, \dots, n \quad (5.19)$$

Lovász Schrijver Lifting

The idea for this third SDP representation of inequality constraints which involve 0/1-variables is to achieve a quadratic form by multiplying the n inequalities (5.16) by either x_i or $(1 - x_i)$, $i=1, \dots, m$. Therefore, for every original

inequality, one gets $2m$ lifted inequalities. To explain this approach in detail we consider the single inequality

$$a^\top x \leq b,$$

where $a \in \mathbb{R}^m$ is a vector and $b \in \mathbb{R}$ is a scalar.

First, multiplying this inequality with $x_i, i = 1, \dots, m$ results in m inequalities:

$$\sum_j^m a_j x_i x_j \leq x_i b \quad \text{for } i = 1, \dots, m \quad (5.20)$$

With $x_i = x_i^2$ for 0/1-variables we obtain for any fixed i the quadratic form:

$$\sum_j^m a_j x_i x_j - b x_i x_i \leq 0 \quad (5.21)$$

Written in trace form, $\text{Tr}[^i A X] \leq 0$, the components of the symmetric constraint matrix $^i A$ are $(k, l = 1, \dots, m)$:

$$^i A_{kl} = \sum_j^n \left(\frac{a_j}{2} (\delta_{ki} \delta_{lj} + \delta_{li} \delta_{kj}) - b \delta_{ki} \delta_{li} \right) = \frac{a_l}{2} \delta_{ki} + \frac{a_k}{2} \delta_{li} - b \delta_{ki} \delta_{li}$$

Secondly, if we multiply the original constraint with $(1 - x_i), i = 1, \dots, m$, a similar calculation leads to m more inequalities:

$$\sum_j^m a_j x_j - \sum_j^m a_j x_i x_j + b x_i \leq b \quad i = 1, \dots, m \quad (5.22)$$

By using $x_j = x_j^2$, we can express this again in trace form $\text{Tr}[^i \hat{A} X] \leq b$. Then the symmetric constraint matrix $^i \hat{A}$ has the following components $(k, l = 1, \dots, m)$:

$$^i \hat{A}_{kl} = \sum_j^n \left(a_j \delta_{kj} \delta_{lj} - \frac{a_j}{2} (\delta_{ki} \delta_{lj} + \delta_{li} \delta_{kj}) \right) + b \delta_{ki} \delta_{li} = a_k \delta_{kl} - \frac{a_l}{2} \delta_{ki} - \frac{a_k}{2} \delta_{li} + b \delta_{ki} \delta_{li}$$

The three liftings discussed above represent standard methods to incorporate linear inequalities into a SDP formulation. We note here that we dropped the linear inequality constraints of the bipartite matching ($\sum_{i=1}^K x_{ij} \leq 1, \forall j$) in our experiments because the SDP solvers we used require equality constraints. At the time of writing, there were no SDP solvers available which are able to cope with the large number of constraints and which are also able to cope with inequality constraints. But the progress in the development of SDP solvers makes it likely that there will be solvers in the near future which also can efficiently cope with inequalities.

In the following we discuss how we obtain appropriate integer solutions from the non-integer solutions of the semidefinite relaxation (5.8).

5.4.4 Post-Processing to obtain an Integer Solution

Once the global optimum $X_{bound} \in \mathbb{R}^{(KL+1) \times (KL+1)}$ of the semidefinite relaxation (5.8) is computed, the diagonal elements of the solution are interpreted as a non-integer solution $\hat{x}_{sol} = \text{diag}(X_{bound})$ for (5.3). Omitting the first element in $\hat{x}_{sol} \in \mathbb{R}^{KL+1}$, which was added due to the homogenization (5.9), we obtain the approximation $x_{sol} \in \mathbb{R}^{KL}$ for the indicator vector $x \in \{0, 1\}^{KL}$. To compute from x_{sol} a 0/1-integer solution which represents an admissible bipartite matching, we solve the following linear program in a post-processing step:

$$\begin{aligned} \max_x & x_{sol}^\top x \\ \text{s.t. } & A_K x = e_K, \\ & A_L x \leq e_L \\ & x \in \{0, 1\}^{KL} \end{aligned} \tag{5.23}$$

As in (5.2), the total unimodular constraint matrices A_K and A_L in (5.23) assure that the optimal solution is a 0/1-integer solution and thus represents a bipartite matching. The only difference to (5.2) is that (5.23) results in a bipartite matching which maximizes the scalar product $x_{sol}^\top x$ instead of minimizing the scalar product $w^\top x$. As a result, the post-processing step (5.23) finds the largest elements in the solution vector x_{sol} which are compatible with a bipartite matching. The 0/1-integer solution obtained represents the approximate solution to the combinatorial optimization problem (5.3).

Based on a probabilistic interpretation of the non-integer solution x_{sol} a more sophisticated post-processing step can be used to obtain a 0/1-integer solution. The probabilistic interpretation and an improved post-processing step are outlined in the following.

Probabilistic Interpretation of the Non-Integer Solution

According to the constraints $A_K x_{sol} = e_K$ each possible match of the model node i to all $j = 1, \dots, L$ scene nodes sums up to one in the solution vector x_{sol} : $\sum_j^L (x_{sol})_{ji} = 1, i = 1, \dots, K$. Using this we interpret the elements $(x_{sol})_{ji}$ of the solution vector probabilistically: The element $(x_{sol})_{ji}$ represents the probability that model node i is matched to scene node j . The overall probability that a model node i is matched to the scene is 1. With this interpretation, the post-processing step (5.23) turns out to select the bipartite matching with the highest total probability and is therefore a reasonable approach to obtain the 0/1-integer solution. Nevertheless, as often a better combinatorial solution exists the probabilistic interpretation of x_{sol} suggests itself to consider the other possible matchings according to their probability as well. To this end, we propose the following post-processing step.

Probabilistically Motivated Post-Processing Step

To exploit the probabilistic information in the non-integer solution vector x_{sol} when computing the 0/1-integer solution, we adopt the well known Metropolis algorithm (see e.g. [107, 144]). Starting from a valid bipartite matching

obtained by (5.23), a node $i \in \{1, \dots, K\}$ of the (smaller) model graph is randomly selected and according to the probabilities in $(x_{sol})_{ji}, j = 1, \dots, L$, a new mapping from i to $j \in \{1, \dots, L\}$ is chosen randomly. If the new matching results in an improved combinatorial solution the solution is accepted. Otherwise the new match is accepted only with a small probability. After a fixed number of iterations the bipartite matching with the smallest objective function is used as final 0/1-approximation. We call one such iteration *sampling step*. In our experiments we have usually stopped the post-processing after $10 \cdot KL$ steps. More iterations surely would increase the probability to obtain better solutions. Note that this approach can not worsen the matching obtained by (5.23). Indeed, it experimentally turns out that this post-processing step often results in improved matchings.

In the following we refer to the SDP relaxation followed by the linear post-processing (5.23) as *basic* SDP subgraph matching approach. The approach followed by the sampling method is called the *probabilistic* or *sampling* SDP subgraph matching approach.

5.5 Illustrative Example

The aim of this section is to illustrate and to clarify the convex subgraph matching approach. To this end, we consider in detail the subgraph matching problem shown in figure 5.1 at the beginning of this chapter. Further experiments are described in section 5.6.

As the direct comparison of our SDP subgraph matching approach with other approaches was not accomplishable, because other approaches have different requirements with regard to the problem data, we have decided to use synthetically created problem instances for our experiments in section 5.6 that are reasonable to simulate object recognition tasks.

5.5.1 The Subgraph Matching Example

For illustrating the main features of our approach, we consider the subgraph matching problem depicted in figure 5.1.

We artificially generated the similarity data w for this subgraph matching example. All assignment costs $w(j, i)$ of the similarity vector w are selected randomly out of different cost ranges, depending on the fact whether the model node i fits to the scene node j or not. We defined three different cost ranges for the possible assignments of the model nodes to the scene nodes:

expensive	0.8 – 1.0
cheap	0.4 – 0.6
very cheap	0.2

In our example the following assignments representing exactly the desired subgraph matching are chosen to be cheap: $1 \mapsto 9$, $2 \mapsto 10$, $3 \mapsto 11$, $4 \mapsto 12$, $5 \mapsto 13$. The other assignment costs are randomly selected out of the expensive range. To make the problem more difficult, the following undesired assignments from model nodes to the background nodes are made very cheap

artificially: $2 \mapsto 1$, $3 \mapsto 8$, $4 \mapsto 4$, $5 \mapsto 2$ (cf. figure 5.1). The exact data for the shown experiment is listed in Appendix B.1.1. The chosen similarity is abstractly represented by the colors of the nodes. A similar color of the model node i and the scene node j indicates that the assignment cost $w(j, i)$ of the assignment $i \mapsto j$ is low.

5.5.2 Linear Bipartite Matching Approach

As discussed in section 5.2.2, the linear bipartite matching approach (5.2) leads to a 0/1-solution which assigns the nodes of the model graph to the locally best fitting scene graph nodes. Any knowledge of the structure is completely ignored by this approach. The resulting undesired matching ($1 \mapsto 9$, $2 \mapsto 1$, $3 \mapsto 8$, $4 \mapsto 4$, $5 \mapsto 2$) for the example problem has already been depicted in figure 5.2.

5.5.3 SDP Subgraph Matching Approach

In contrast to the linear optimization approach (5.2) our convex programming approach (5.8) followed by the post-processing step (5.23) is able to find the desired bipartite matching. The resulting matching, represented by the red line segments, is shown in figure 5.8. Furthermore, figure 5.8 shows some

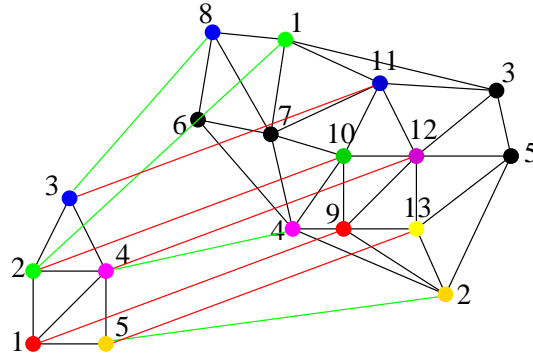


Figure 5.8: The SDP approach leads to the desired bipartite matching shown by the red line segments. Furthermore, probable candidate matchings are depicted with green lines.

candidate mappings (green lines) which are obtained by inspecting the non-integer approximation vector x_{sol} computed for the example problem. Recall that x_{sol} is equal to the diagonal of the solution matrix X_{sol} (omitting the first element) of the SDP problem (5.8), and that x_{sol} is used to obtain the binary integer solution.

Figure 5.9 shows the approximation vector x_{sol} plotted against its index. The plot is subdivided into $K = 5$ segments, with the i -th segment ($i \in \{1, \dots, K\}$) representing all possible matchings from the model node i to all $L = 13$ nodes in the scene graph (cf. section 2.2.2).

Corresponding to the constraints $A_K x_{sol} = e_K$, one can verify that each segment sums up to 1. Therefore, as discussed in section 5.4.4, we may interpret the elements $(x_{sol})_{ji}$ of x_{sol} as the likelihood that model node i is mapped to scene

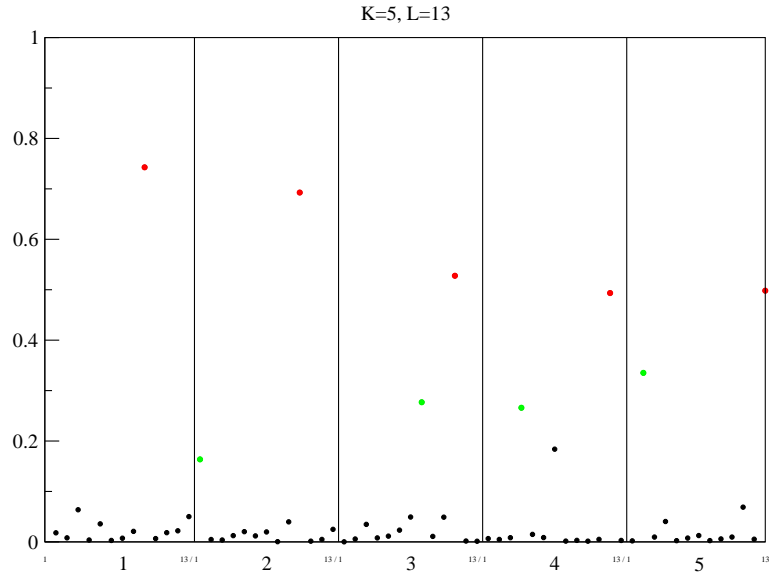
Solution Vector x_{sol} for the Example Subgraph Matching Problem

Figure 5.9: Non-integer solution related to the subgraph example experiment. The solution vector x_{sol} ($\alpha = 0.25$) is plotted against its index. The most probable matching, obtained by the post-processing step (5.23), is marked red. Other reasonable mapping candidates are marked green. The corresponding matchings are shown in figure 5.8.

node j . The solution shown supports this clearly, as only reasonable mappings $i \mapsto j$ have higher values $(x_{sol})_{ji}$ in the solution vector x_{sol} . For example, in the second segment (see figure 5.9) only two reasonable mappings $2 \mapsto 1$ and $2 \mapsto 10$ have significantly large values which are represented by the vector elements $x_{1,2} \approx 0.16$ and $x_{10,2} \approx 0.69$. The mapping $2 \mapsto 10$, which is indeed part of the optimal matching, can be seen as more likely than the mapping $2 \mapsto 1$ which is one of the cheap mappings.

To obtain the bipartite matching from the approximation vector x_{sol} , the post-processing (5.23) is used to calculate a nearby 0/1-integer vector. The post-processing selects in each segment the highest possible, and in our interpretation the most likely entry, which is compatible with a bipartite matching. The most likely entries are marked red, and the resulting closest 0/1-integer vector is shown in figure 5.10. It represents indeed the global optimum of (5.3). Other

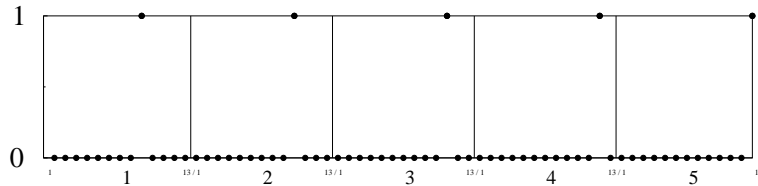


Figure 5.10: Integer solution representing the obtained bipartite matching for the small subgraph matching example. It is obtained by the post-processing step (5.23) from the non-integer solution x_{sol} which is shown in figure 5.9.

probable candidates are marked green in figure 5.9. The dot colors are in accordance with the colors of the matching and candidate mappings shown in figure 5.8.

5.5.4 Problem Nature

To visualize the character of the subgraph matching problems, the probability distribution of the objective values (5.9) for the small subgraph matching problem is shown in figure 5.11. The parameter α was set to 0.25. Our small

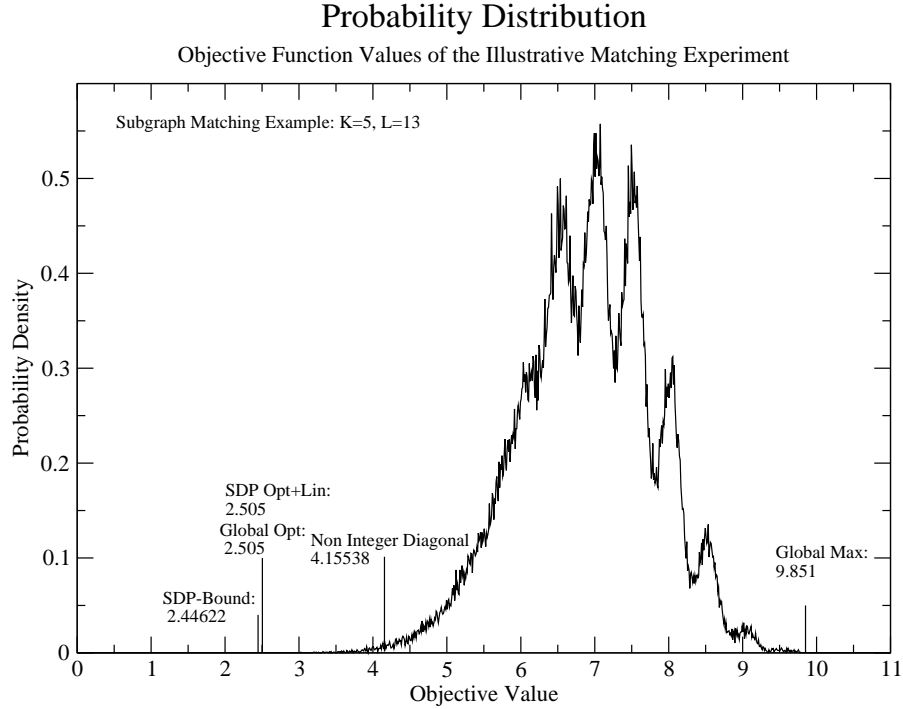


Figure 5.11: Distribution of the objective values of the illustrative graph matching problem. The shown lower bound (2.44622) is obtained by the SDP solution of (5.8). The global optimum (2.505) was found by the graph matching approach. Additionally the objective value (4.15538), obtained by using the diagonal vector in (5.9) of the SDP solution, is shown (cf. figure 5.8).

example with $K = 5$ model nodes and $L = 13$ scene nodes has $\frac{L!}{(L-K)!} = 154440$ possible assignments. According to figure 5.11 a *randomly* selected assignment is likely to result in an objective value around 7.0 ± 1.0 . But the aim is to find the assignment that results in the global minimum which is 2.505 (cf. (5.3)). The SDP relaxation (5.8) results in a very good lower bound 2.44622 close to the global optimum. The tightness of the bound is a further indicator that our approach is capable to find good approximations to the combinatorial original problem. The non-integer approximation vector x_{sol} leads to an objective value (cf. (5.9)) of 4.15538 which is to some extent away from the global minimum.

However, x_{sol} contains sufficient information to obtain the optimal solution by the post-processing step (5.23).

This small subgraph example shows very clearly the capability of the SDP graph matching approach to find good subgraph matchings.

In the following sections, we investigate in detail the SDP subgraph matching approach in order to reveal its performance and applicability for computer vision tasks. We present statistical results computed for large sets of randomly generated subgraph matching problems. The experimental results are very promising but they also reveal the difficult combinatorial nature of subgraph matching problems.

5.6 Subgraph Matching Experiments

In the following we present results for subgraph matching experiments which are supposed to simulate a broad range of subgraph matching problems where we assume that the object graph is present in the scene graph. After first describing in section 5.6.1 how the problem instances were created, we show in section 5.6.2 for a single but representative problem instance the objective value distribution along with results obtained by our SDP subgraph matching approach. In section 5.6.3 we then discuss the typical influence of the parameter α considering a particular subgraph matching problem instance. After guessing the value of α we show in section 5.6.4 several statistical performance results of the SDP subgraph matching approach for a large series of experiments with increasing problem size.

5.6.1 Creation of the Problem Instances

Each problem instance for the experiments discussed in this section was created as follows: First, the object graph with K nodes is randomly created with an edge probability² of 0.5. Then the scene graph is created by copying the object graph and enlarging it to L nodes by adding $(L - K)$ random nodes and edges with an edge probability of 0.2. An example for such an object and scene graph structures is shown in figure 5.12. One can see that the object graph structure is present in the scene graph. To compute the similarity between the nodes of the two graphs we defined the following two overlapping cost ranges:

small range 0.4 – 0.6

wide range 0.4 – 1.0

The cost w_{ji} , $i = 1, \dots, K$, $j = 1, \dots, L$ is selected randomly within the small range (cheap) if the mapping of the object node i to the scene node j represents a desired mapping. Otherwise the cost is set randomly to a value within the wider range. Note that the wider range also includes the small range which makes it likely that some undesired mappings have cheaper assignment costs w_{ji} than the appropriate desired mapping. Therefore, the linear matching approach (5.2), which ignores the graph structure, is likely to fail in all experiments.

²The edge probability is the probability that an edge of the underlying complete graph is present.

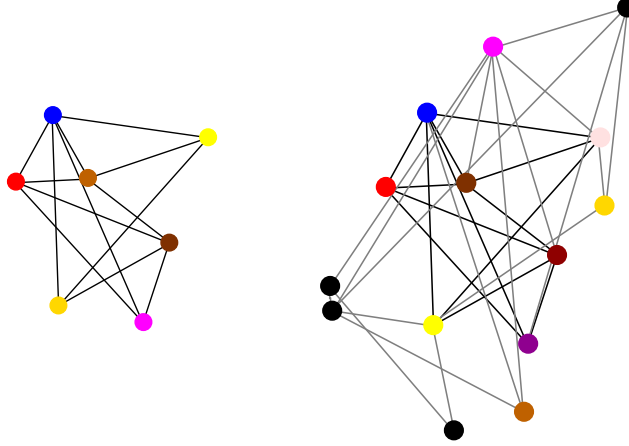


Figure 5.12: Illustration of a randomly created subgraph matching problem. The object graph structure (left) is, due to the creation process, part of the scene graph structure (right). The cost for a desired mapping is low, while for an undesired mapping the cost may be any value including low ones. Here, a similar color of an object node and a scene node indicates a low assignment cost.

However, due to the problem creation we know where the object graph is present in the scene graph and we are able to estimate the optimal solution of (5.3). Our guess represents the desired matching which maps the object graph to the object graph structure within the scene. For these estimated solutions x_{est}^* the quadratic term in (5.3) is zero as the object graph structure is mapped isomorphically to the subgraph in the scene. We have observed that in most cases this estimation coincides with the optimal solution. But it should be noted that in some rare cases an undesired matching can have a better objective value. Nonetheless, we regard these experiments as ground truth experiments because in nearly all cases our guess represents the optimal solution.

Abbreviations

We summarize here some notations which are used within the following sections. We denote the lower bound obtained by the SDP relaxation (5.8) with $f_{bound}^* \in \mathbb{R}$ and the appropriate solution matrix with $X_{bound}^* \in \mathbb{R}^{(KL+1) \times (KL+1)}$. Recall that the non-integer solution vector $x_{sol} \in \mathbb{R}^{KL}$ is basically the diagonal³ of X_{bound}^* . The estimated optimal combinatorial optimum will be denoted with $x_{est}^* \in \{0, 1\}^{KL}$ and the corresponding objective value with $f_{est}^* \in \mathbb{R}$. Similarly, if the exact global minimum of (5.3) is known we refer to its mapping as $x_{opt}^* \in \{0, 1\}^{KL}$ and to the objective value as $f_{opt}^* \in \mathbb{R}$. Furthermore, we denote the solutions which are obtained by the basic SDP subgraph matching approach with $x_{lin}^* \in \{0, 1\}^{KL}$ and the solution computed by the sampling SDP subgraph matching approach with $x_{sampling}^* \in \{0, 1\}^{KL}$. The corresponding objective

³Omitting the first element

values are named analogously $f_{lin}^* \in \mathbb{R}$ and $f_{sampling}^* \in \mathbb{R}$.

5.6.2 Problem Nature

In order to see the nature of the combinatorial optimization problem (5.3) for a particular but representative subgraph matching problem instance the probability distribution of the objective values is drawn in figure 5.13. The object and scene graph size for this problem instance are $K = 9$ and $L = 22$, respectively. Due to the high combinatorial number of possible mappings ($\approx 1.8 \cdot 10^{11}$)

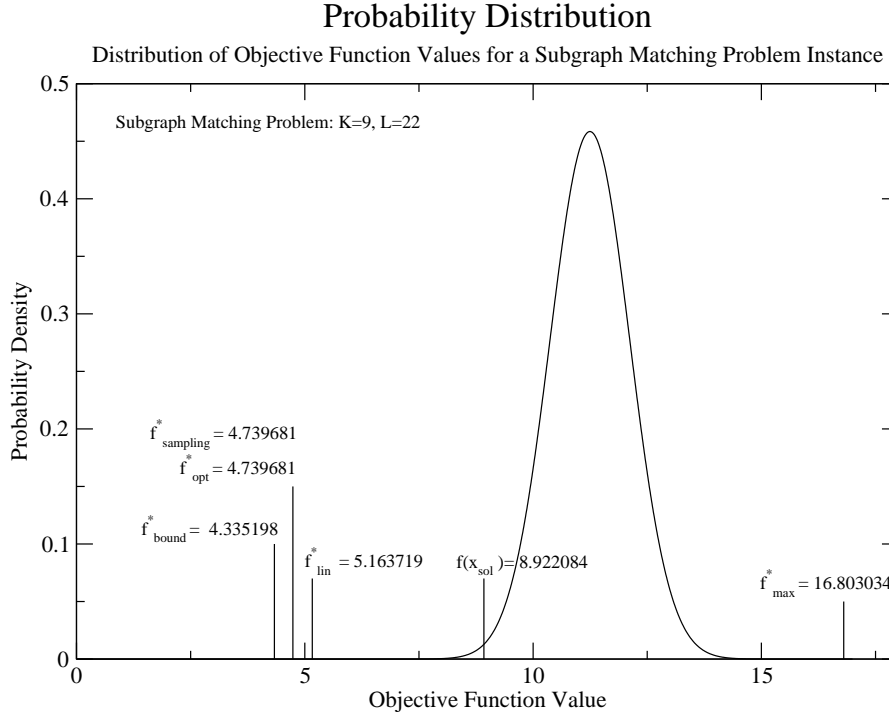


Figure 5.13: Probability distribution of the objective values for a specific but representative subgraph matching problem instance with $K = 9$ and $L = 22$. The calculated lower bound f_{bound}^* is near to the global optimum f_{opt}^* which indicates that the relaxation is quite tight. The global optimum was obtained by the sampling post-process and we have $f_{sampling}^* = f_{opt}^*$.

the distribution looks much smoother than the distribution for the small subgraph matching problem shown in figure 5.11. The aim of the optimization problem (5.3) is to obtain the matching with the smallest objective value f_{opt}^* . Along with the probability distribution some computationally relevant values like the lower bound f_{bound}^* and the objective values f_{lin}^* and $f_{sampling}^*$ of the computed 0/1-integer solutions are shown. One can see that the lower bound is near the global optimum which indicates that the relaxation for this problem is quite tight. The basic SDP subgraph matching approach results in an objective value f_{lin}^* higher than the global optimum f_{opt}^* but the sampling post-processing step reveals the global minimum and we get $f_{sampling}^* = f_{opt}^*$. The

graph structures of this particular subgraph matching instance are depicted in figure 5.14. Furthermore, the optimal mapping $x^* \in \{0, 1\}^{KL}$ is shown by red

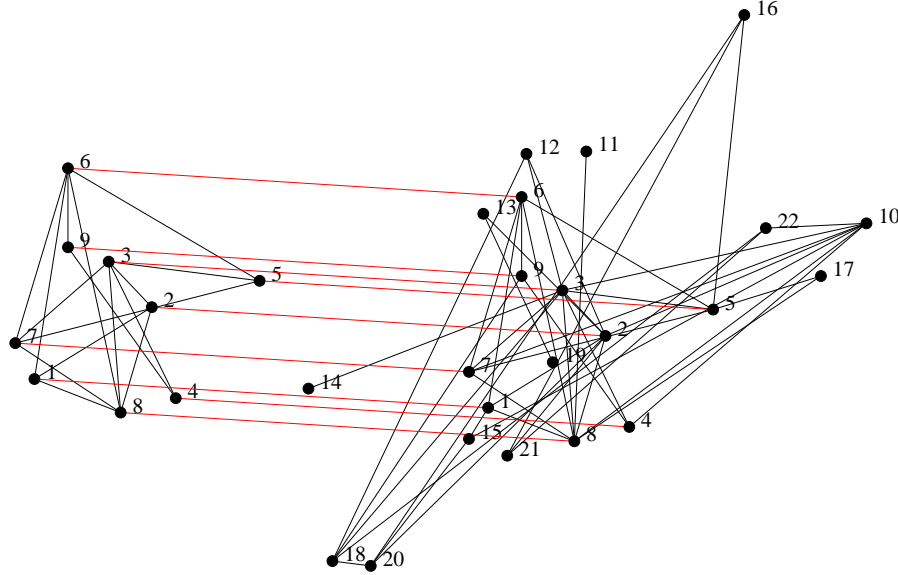


Figure 5.14: The shown graph structures belong to the subgraph matching problem instance for which the probability distribution of objective values is shown in figure 5.13. The optimal mapping $x_{opt}^* \in \{0, 1\}^{KL}$ represents also the desired mapping and is indicated by red lines.

lines and we can see that it represents also the desired mapping. Therefore, the problem instance discussed here supports our assumption that we can search for the global optimum of (5.3) to obtain a desired subgraph matching.

5.6.3 Influence of the Regularization Parameter

The factor α is the only approach inherent parameter which has to be adjusted in our SDP subgraph matching approach. To understand its influence we next discuss the parameter dependency considering a small subgraph matching problem which shows the typical behavior. Then we sketch how we chose the parameter for our experiment series considered in section 5.6.4.

Typical Parameter Dependency

For a small subgraph matching problem with $K = 6$ object nodes and $L = 12$ scene nodes the parameter dependency of the combinatorial optimum f_{opt}^* (green) along with the lower bound f_{bound}^* (black) computed by the relaxation (5.8) is shown in figure 5.15. The exact problem data can be found in Appendix B.1.2.

The global optimum $f_{opt}^*(\alpha)$ for $\alpha = 0$ is identical with the optimum obtained by the linear matching approach (5.2). From this point the global optimum starts to increase linearly with α . The slope of $f_{opt}^*(\alpha)$ is proportional to the

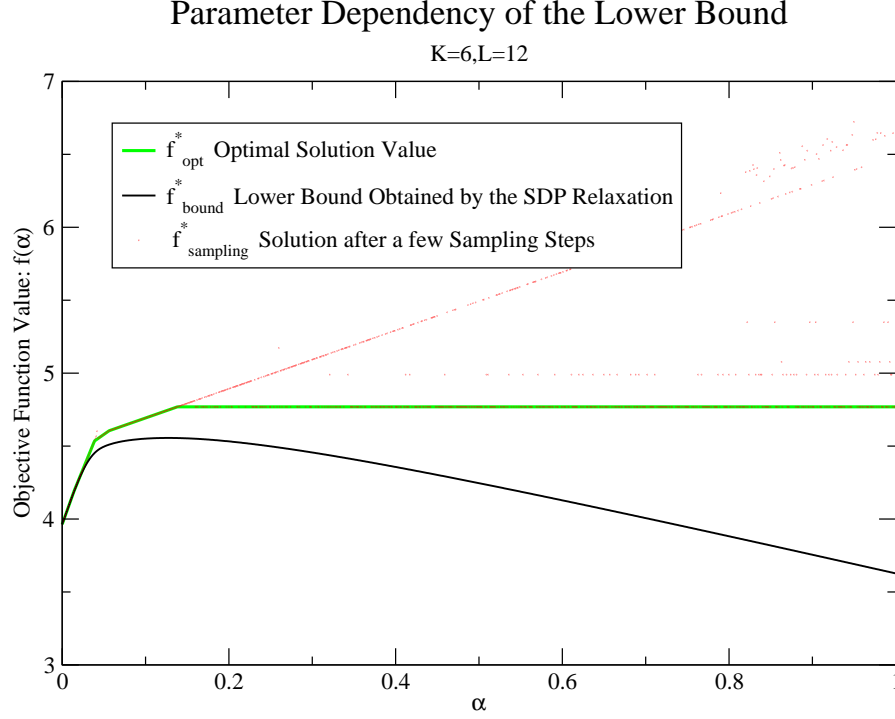


Figure 5.15: Parameter dependency of the global minimum $f_{opt}^*(\alpha)$ (green) of the combinatorial subgraph matching approach (5.3) for a typical subgraph matching instance with $K = 6$ and $L = 12$ along with the lower bound $f_{bound}^*(\alpha)$ (black) obtained by the SDP relaxation (5.8). Moreover, the objective values $f_{sampling}^*$ (red dots) which are obtained by the sampling post-processing are shown.

number of structural differences in the obtained matching. With every change of this slope the combinatorial optimum $x_{opt}^* \in \{0, 1\}^{KL}$ changes to a mapping with less structural differences. We see that with increasing α solutions with less structural differences are preferred and for the discussed subgraph matching problem instance we can obtain a full desired matching when setting $\alpha > 0.14$. At about this α the slope of $f_{opt}^*(\alpha)$ reaches zero from which we conclude that the corresponding mapping x_{opt}^* represents a subgraph isomorphism of the two graph structures as the quadratic term in (5.3) must be zero. The graph structure of the discussed example along with the optimal and also desired mapping for $\alpha > 0.14$ is shown in figure 5.16.

The lower bound $f_{bound}^*(\alpha)$ follows for small α 's smoothly the characteristics of the combinatorial global optimum. It attains a maximum at $\alpha \approx 0.13$ after which a slow nearly linear decrease starts. We observe that with increasing α the gap between the optimal solution $f_{opt}^*(\alpha)$ and the lower bound $f_{bound}^*(\alpha)$ increases, which indicates that the approximation gets less tight for increasing α . The reason for the decreasing lower bound lies in the high degree of freedom for the positive semidefinite matrix X in the relaxation (5.8). There can be

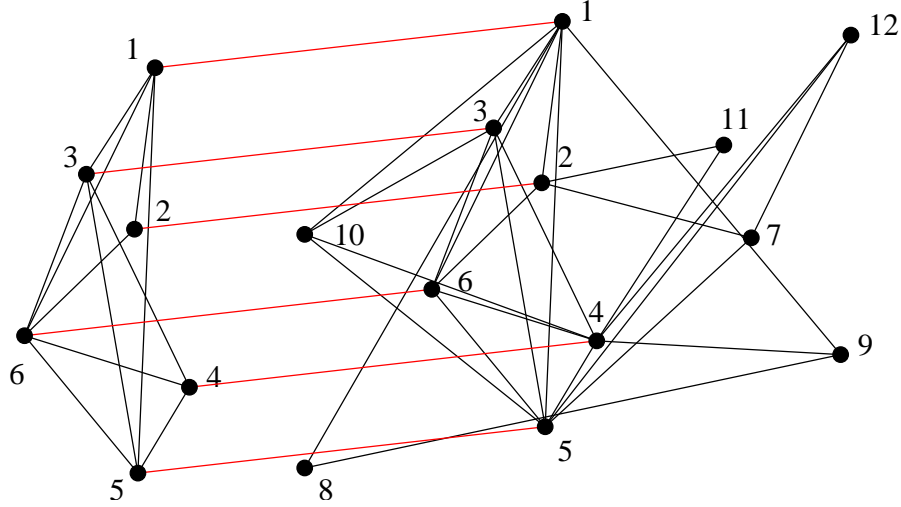


Figure 5.16: Graph structure of the subgraph matching problem instance used to visualize the typical parameter dependency. The shown desired mapping (red line segments) is equivalent to the optimal mapping for $\alpha > 0.14$.

negative elements in X which squeeze down the objective value of the relaxation without affecting the positive semidefiniteness of the matrix X .

Furthermore, in figure 5.15 the objective values $f_{sampling}^*(\alpha)$ obtained after $10 \cdot KL$ iterations of the sampling step are shown by red dots. In many cases the global optimum is achieved by this probabilistic post-processing. We note that more iterations of the sampling step would yield the global optimum more often. The chosen number of iterations reveals some other interesting facts about this specific problem: First, there are at least three other matchings which also correspond to a subgraph isomorphism of the graph structures. In these cases the dots of $f_{sampling}^*(\alpha)$ lie on a horizontal line. The subgraph isomorphism corresponding to the objective value $f_{sampling}^*(\alpha > 0.14) \approx 5.0$ is shown in figure 5.17.

Usually problems of this size are solved to optimality already by the basic SDP subgraph matching approach. But due to the structural ambiguity the optimal mapping is less clear present in x_{sol} and can only be obtained by the sampling post-processing step. Secondly, there are many solutions which lie on straight lines with a slope of 2. These are mappings where just one edge must be added or removed from the object graph structure to obtain a subgraph isomorphism between the object and the scene graph structure (cf. section 5.3.1). Thirdly, it can be seen that with an increasing gap between the lower bound and the optimum more often different solutions $f_{sampling}^*$ and with this different mappings $x_{sampling}^* \in \{0, 1\}^{KL}$ are obtained by the sampling post-process. That shows that the tightness which measures the distance of the lower bound to the optimum affects the ability to compute the optimal combinatorial solution by the sampling post-processing step.

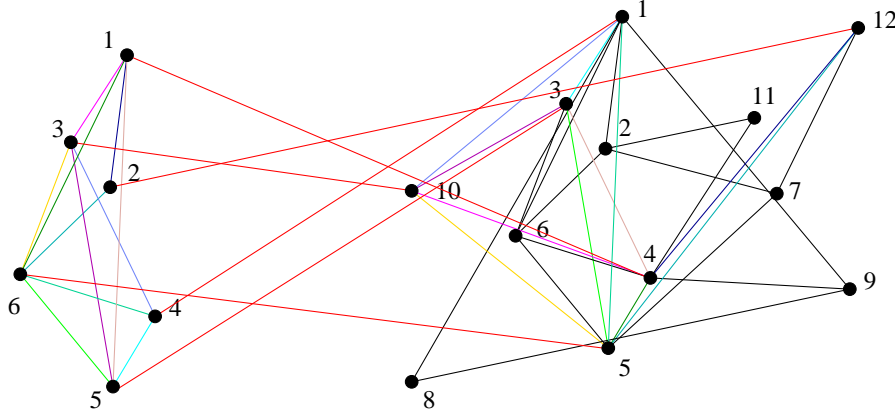


Figure 5.17: A possible matching which represents also a subgraph isomorphism between the object graph structure and the scene graph structure for the here discussed problem instance. The mappings of the nodes are depicted by the red lines and corresponding edges are marked by the same color.

Choice of the Parameter

To obtain a reasonable estimate for the regularization parameter α for the experiment series which we will discuss in section 5.6.4 we investigate here the parameter dependency of our SDP subgraph matching approach for one large set of 1000 problem instances with the size $K = 9$ and $L = 25$. We note that the problem instances were not changed with varying regularization parameter. Generally, if α is too small the regularization is not able to correct an undesired matching which is obtained by the linear matching approach (5.2). On the other side, if α is too large the relaxation gets less tight and good solutions become less likely. Therefore one has to find a compromise for α that is large enough to result in a desired matching but that also results in a tight relaxation. However, it turns out that the sampling SDP subgraph matching approach is not very sensitive to the choice of the parameter values. This can be seen in figure 5.18, where we show the percentage of problem instances which result in the estimated or even better optimum.

For the investigated problem instances the basic SDP subgraph matching approach reaches a maximum fraction of 38.3% optimal solutions for $\alpha \approx 0.15$ which decreases to 25.5% for $\alpha \approx 0.5$. The additional sampling post-processing improves this fraction to a much better rate of over 80% which is not very sensitive to the choice of α . This supports our assumption that the non-integer solution x_{sol} comprises information that can be statistically exploited to obtain good integer solutions.

From the viewpoint of computer vision an important measurement is the number of nodes one can expect to be mapped in accordance with the desired mappings. Therefore, we plotted in figure 5.19 the mean number of computed mappings which are in accordance with the desired mappings for increasing

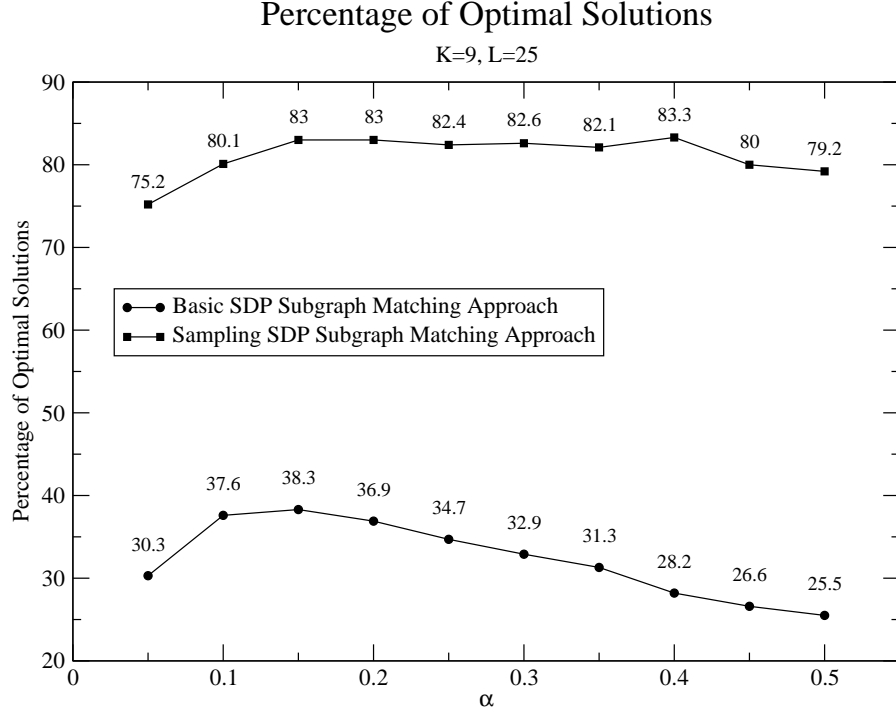


Figure 5.18: Percentage of computed solutions with an objective value equal or better than the estimated solution. The sampling significantly improves the fraction of optimal solution obtained by the basic SDP subgraph matching approach.

parameter α . As we have fixed the number of object graph nodes to $K = 9$ in our experiments, there are exactly 9 desired mappings. We can see that for $\alpha = 0$ (the linear case) one can expect to obtain only approximately 1 desired mapping (red dot). From $\alpha = 0.1$ to $\alpha = 0.5$ our basic approach results quite stably in about 7 (out of 9) desired mappings which is further increased to 8 (out of 9) by the sampling post-processing step. Considering the large amount of possible matchings ($7.4 \cdot 10^{11}$), it is promising that the sampling SDP subgraph matching approach can be expected to obtain approximately 8 of the 9 mappings correctly for the considered problem instances with $K = 9$ and $L = 25$. Therefore, this measurement verifies that our SDP approach is capable to compute matchings that are consistent with desired matchings.

Based on these experiments we have chosen $\alpha = 0.15$ for the experiment series in section 5.6.4 as it maximizes the fraction of globally optimal solutions obtained by the basic SDP subgraph matching approach and results in a high number of desired mappings for this experiment. Putting everything together, our experiments revealed that reasonable values for α are between $\alpha = 0.01$ and $\alpha = 0.40$ depending on the problem data⁴.

⁴ We assume that the similarity measure has values in the range between 0.0 and 1.0.

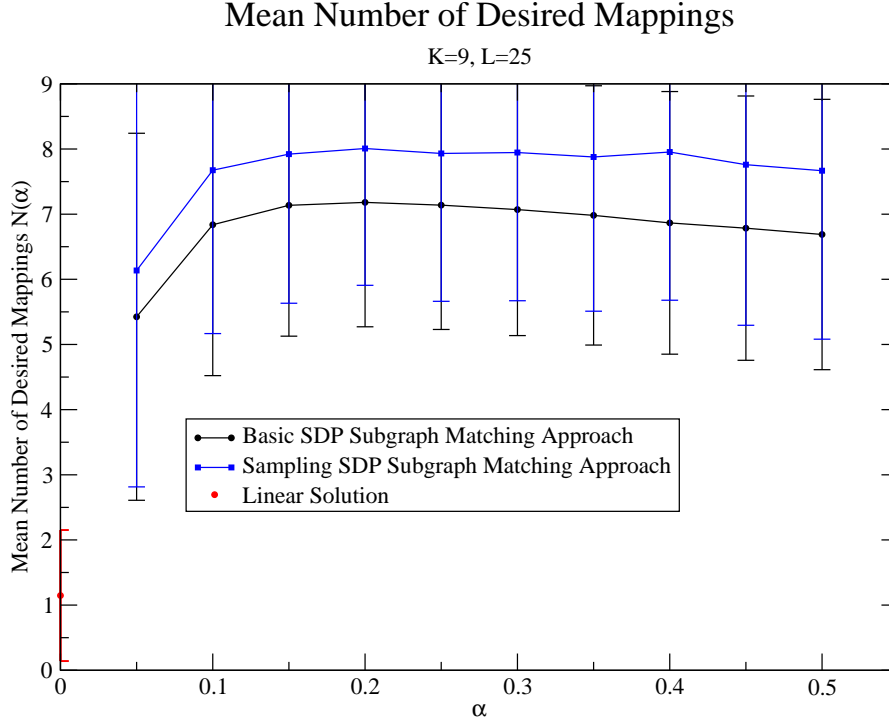


Figure 5.19: The expectation value for the number of nodes which are mapped in accordance with the desired mappings is not very sensitive to the parameter α . The SDP subgraph matching approach, followed by the sampling post-process, can be expected to result in nearly 8 (out of 9) correctly detected desired mappings for the considered problem instances ($K = 9, L = 25$).

5.6.4 Statistical Performance Investigation

In this section we investigate statistically the capacity of the SDP subgraph matching approach in terms of the problem size. To this end we study several performance measurements for large sets of subgraph matching problems with increasing problem size. In particular we investigate the tightness of the lower bound, the percentage of optimally solved problem instances, the mean ratio of the computed combinatorial objective value to the estimated optimum and the number of correctly detected desired mappings. The problem instances are created as described in section 5.6.1 and we set the object graph size to $K = 9$ while we increased the size of the scene graph from $L = 14$ to $L = 30$. For every problem size we created 1000 problem instances.

Recall that the number of possible assignments in the subgraph matching experiments is $L!/(L - K)!$. This exponential growth is depicted in figure 5.20.

Tightness

If the lower bound f_{bound}^* which was obtained by the SDP relaxation (5.8) is close to the global optimum f_{opt}^* then the relaxation is called to be tight. Therefore

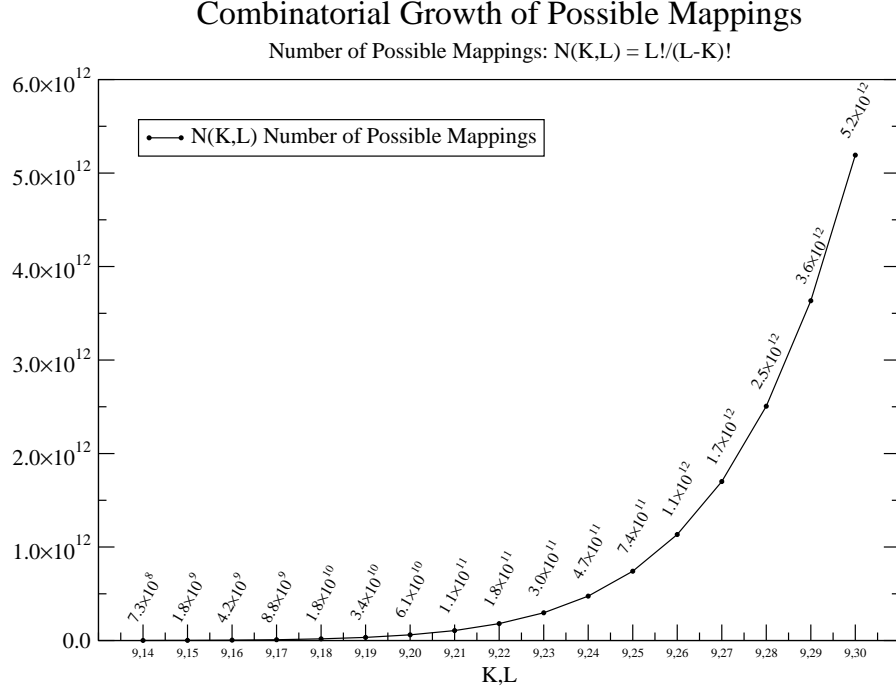


Figure 5.20: Combinatorial growth of possible assignments for subgraph matching problems with $K = 9$ and increasing scene graph size from $L = 14$ to $L = 30$.

we compare here the lower bound f_{bound}^* with the estimated optimum f_{est}^* to have a direct measurement of the tightness. In particular we used the measure $T(f_{est}^*, f_{bound}^*) = (f_{est}^* - f_{bound}^*)/f_{est}^*$ which tends to be zero if the relaxation is very tight. In figure 5.21 the mean value of the tightness measure $T(f_{est}^*, f_{bound}^*)$ is plotted and we can see that for the problems of smaller size the relaxation is usually very tight as the mean of $T(f_{est}^*, f_{bound}^*)$ is very close to zero. For the larger problems the relative distance between the lower bound and the estimated optimum increases. In the following we will see how the decreasing tightness influences some other performance measures.

Fraction of Optimally Solved Problem Instances

To get an idea of the capability of our subgraph matching approach we investigated the fraction of optimally⁵ solved problem instances. The results for the here discussed experiment series obtained by the SDP subgraph matching approach are plotted in figure 5.22. We compare the percentage of optimal solutions obtained by the basic SDP subgraph matching approach with the percentage of optimal solutions obtained by the sampling SDP subgraph matching approach. For smaller problems in nearly all cases the global optimum is obtained by the basic SDP subgraph matching approach, which was

⁵ Note that we count a solution as optimal if it has an equal or better objective value than the objective value f_{est}^* of the estimated solution.

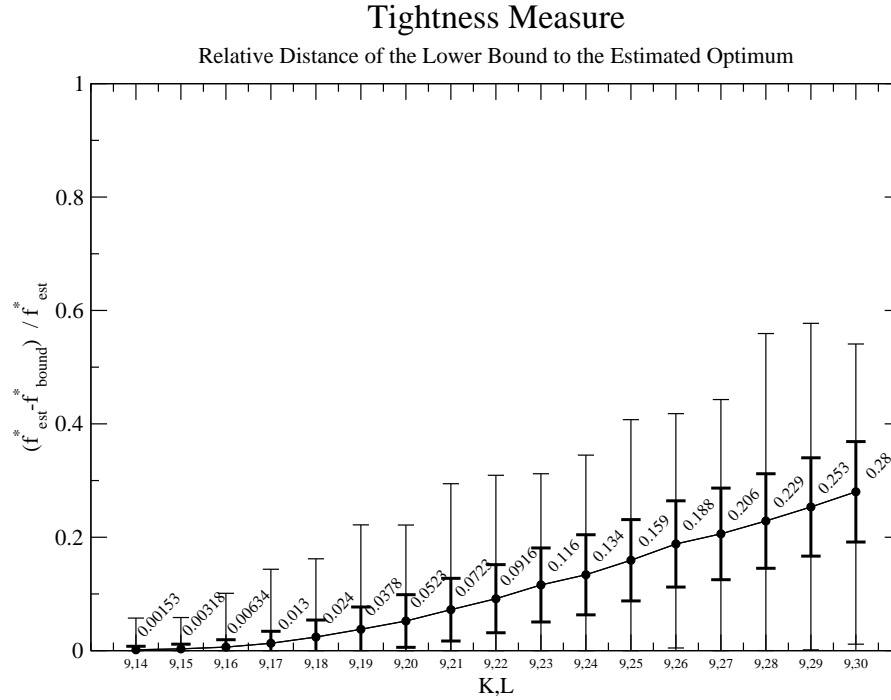


Figure 5.21: The tightness measure $T(f_{est}^*, f_{bound}^*) = (f_{est}^* - f_{bound}^*) / f_{est}^*$ indicates that the relaxation is very tight if it is close to zero. We can see that for smaller problem instances the relaxation is usually very tight but with increasing problem size the relaxation becomes less tight.

further improved by the sampling post process. With increasing problem size the percentage of optimal solutions obtained by the basic SDP subgraph matching approach decreases to about 13% for problems with a scene graph size of $L = 30$. However, even for the larger sized problems the probabilistic sampling was able to increase the fraction of optimal solutions to a high rate of about 70% optimally solved problem instances. This indicates that a probabilistic interpretation of the non-integer solution vector x_{sol} obtained by the SDP relaxation is highly reasonable.

Relative Optimum

To get a more accurate picture of the performance of our SDP subgraph matching approach we investigated the quality of the combinatorial solutions. Therefore, we computed the mean values of the ratios f_{lin}^* / f_{est}^* and $f_{sampling}^* / f_{est}^*$ for the solutions f_{lin}^* and $f_{sampling}^*$ obtained by the basic and sampling SDP subgraph matching approach, respectively. A ratio near to 1 indicates that the obtained solution is close to the estimated optimum f_{est}^* .

In figure 5.23 the mean of the ratio f_{lin}^* / f_{est}^* is shown. Beside the mean values of these ratios the standard deviation and the corresponding minimal and maximal values are shown in the figure. We can see that for smaller problems

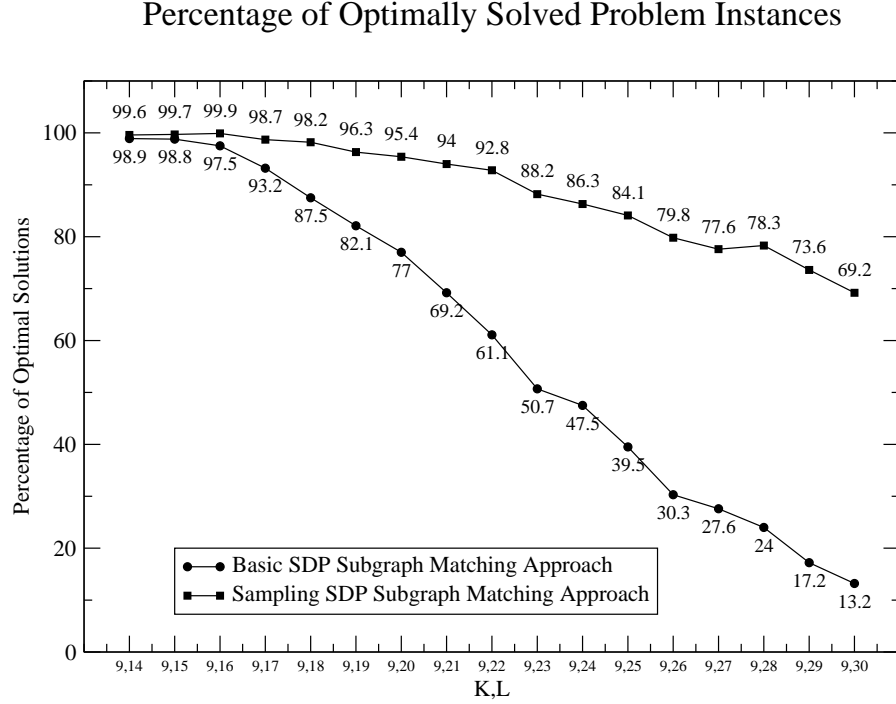


Figure 5.22: Fraction of optimally solved problem instances for increasing problem sizes. The sampling post-process is able to strongly increase this fraction by exploiting the information in the non-integer solution vector x_{sol} .

nearly always the optimal solution is obtained, but with increasing problem size the solution f_{lin}^* obtained by the basic SDP graph matching approach is on the average up to 35% above the estimated global minimum f_{est}^* .

In figure 5.24 the mean value of the ratio $f_{sampling}^*/f_{est}^*$ is shown. One can see that the sampling highly improves the results obtained by the basic approach and draws in the mean the solution $f_{sampling}^*$ very near to its estimated optimal objective value f_{est}^* . Also, for larger problems ($K = 9, L = 30$) the sampling solution $f_{sampling}^*$ is usually only 5% away from the estimated optimum f_{est}^* . We note that the ratio can become smaller than 1 as sometimes the obtained objective value was smaller than the estimated optimum.

Desired Mappings

In order to analyze the applicability of the SDP subgraph matching approach for subgraph matching problems which may occur in computer vision problems we have also investigated the number of correctly detected desired mappings that one can expect to obtain. We have plotted in figure 5.25 the mean number of correctly detected desired mappings obtained by our SDP subgraph matching approach. We can see that the bipartite matching approach (5.2) leads on average to only 2 desired mappings for the small subgraph matching problems

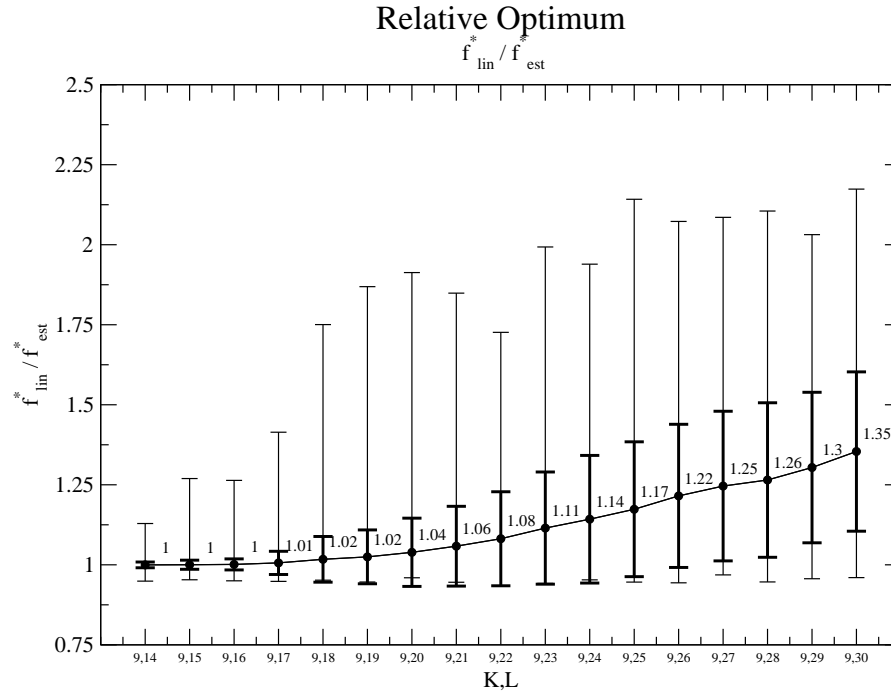


Figure 5.23: Mean ratio f_{lin}^* / f_{est}^* of the objective value f_{lin}^* obtained by the basic SDP graph matching approach and the estimated optimal objective value f_{est}^* . A ratio value near to 1 indicates that the obtained solution is close to the estimated solution.

and decreases further to only 1 desired mapping for the larger problems. The basic SDP subgraph matching approach can be expected to result nearly always in the desired mapping for the small problem instances which is consistent with the observation that nearly always the optimal objective value was obtained. Therefore this is an experimental verification that the optimal solution is usually consistent with the desired matching. With increasing problem size we observe that the mean number of correctly detected desired mappings obtained by the basic SDP graph matching approach decreases to about 5.5 for problem instances with the size $K = 9$ and $L = 30$, but the sampling post-process is able to increase this rate to 7 out of 9 desired mappings.

Summary

We have seen in this section that the performance of the SDP subgraph matching approach depends largely on the size of the subgraph matching problem instances. The tightness which can be seen as a measure for the ability of the relaxation to approximate the original problem, decreases slowly with increased combinatorial problem size. For smaller problems the SDP relaxation is usually tight enough that the simple post-processing step results in the combinatorial optimum. But with the decreasing tightness (the tightness measure T increases)

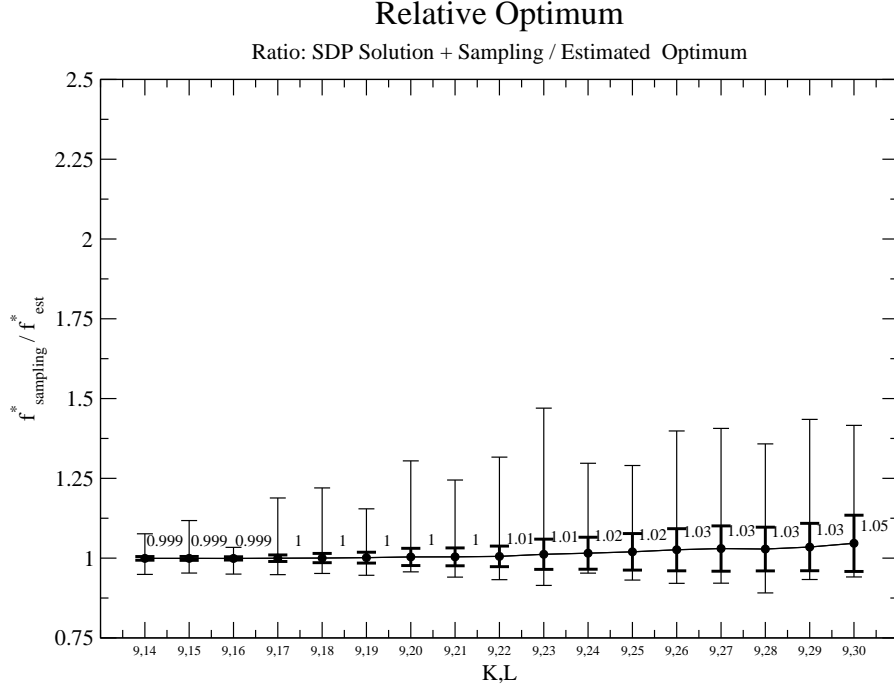


Figure 5.24: Mean ratio $f_{\text{sampling}}^* / f_{\text{est}}^*$ of the objective value f_{sampling}^* obtained by the sampling SDP subgraph matching approach to the estimated optimum f_{est}^* . The results are significantly improved compared to the mean ratio $f_{\text{lin}}^* / f_{\text{est}}^*$ shown in figure 5.23.

the ability of the SDP subgraph matching approach to optimally solve problem instances decreases too. Nevertheless, keeping the combinatorial nature of these problems in mind the results are very promising and even for larger problems very good combinatorial solutions are computed by our sampling SDP subgraph matching approach. In the next section we will see that also the kind of the subgraph matching problems is an important factor for the performance of the SDP subgraph matching approach.

5.7 Random Subgraph Matching Experiments

In this section we investigate the application of our SDP subgraph matching approach to problem instances which are supposed to simulate subgraph matching problems where the object graph is not present in the scene graph. Our aim is to compare the obtained results for these problem instances with the solutions for the problem instances in the previous section. We find that the more random subgraph matching problem instances, investigated in this section, are more difficult to solve. We describe in section 5.7.1 the creation of the here discussed problem instances and show in section 5.7.2 the problem nature of a single but representative problem instance. Then we investigate in section 5.7.3 statistically the performance of our SDP subgraph matching approach for large

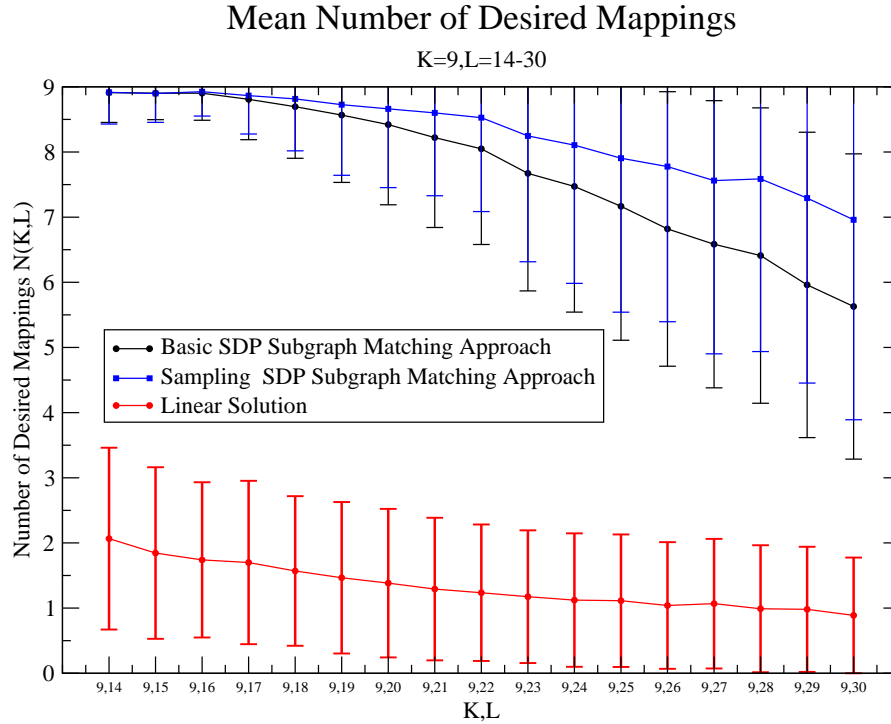


Figure 5.25: Mean number of correctly detected desired mappings obtained by the SDP subgraph matching approach. Also for larger problem instances with $K = 9$ and $L = 30$ the sampling SDP subgraph matching approach can be expected to obtain 7 out of 9 desired mappings correctly.

sets of problem instances.

5.7.1 Creation of the Problem Instances

For the experiments we discuss in this section we have created problem instances where the object and scene graphs are created randomly and independently from each other. Thus, this time, we are not able to guess an optimal assignment. To compute the ground truth for these problem instances by exhaustive search in a reasonable amount of time we set the size of the object graph to $K = 9$ while we let the scene graph size vary only from $L = 14$ to $L = 19$. The vector elements w_{ji} , $i = 1, \dots, K$, $j = 1, \dots, L$ of the similarity measure w are set randomly out of the range $0.4 - 1.0$. Therefore this time there is no correlation between the similarity and the structure as there was in the previous section. We set the edge probability of the (smaller) object graph to 0.4 and the edge probability of the scene graph to 0.3. In figure 5.26 we show two such randomly created graph structures which belong to the set of problem instances where the object graph has $K = 9$ and the scene graph has $L = 19$ nodes. For the experiments discussed here the parameter α was set to 0.3.

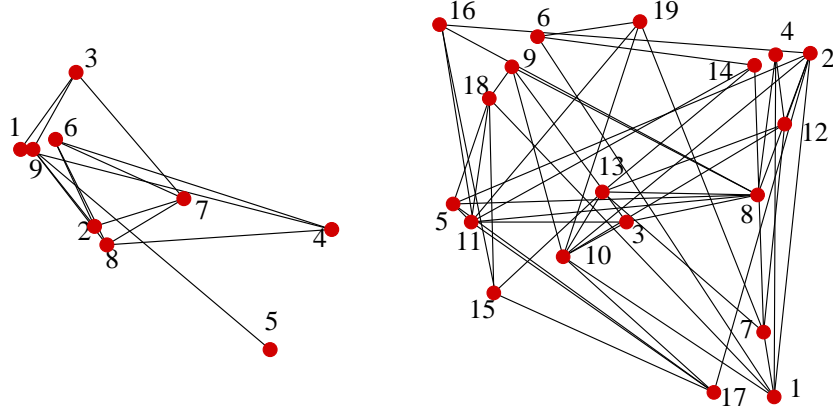


Figure 5.26: Example for the graph structures of a subgraph matching problem instance where the object and scene graph are created randomly and independently from each other. The object graph (right) with $K = 9$ was created randomly with an edge probability of 0.4 and the scene graph (left) with $L = 19$ was created independently with an edge probability of 0.3. The not depicted similarity measure w between the nodes of the two graphs has random values out of the range 0.4–1.0.

5.7.2 Problem Nature

For a single but representative subgraph matching problem instance the probability distribution of the objective values along with some computational results obtained by our subgraph matching approach is depicted in figure 5.27. The object and scene graph structure which belongs to this particular problem instance are shown in figure 5.26. The main difference to the results shown in figure 5.13 is that this time the lower bound f_{bound}^* is not as close to the global optimum f_{opt}^* as the lower bound was to the estimated optimum f_{est}^* in figure 5.13. That indicates a lower tightness of the SDP relaxation for this kind of problem instances and will be supported statistically by the results in section 5.7.3. Due to the lower tightness the optimal combinatorial solution is less likely obtained by the sampling post-processing step. Nevertheless, in terms of the objective value a good combinatorial solution is obtained by the sampling as $f_{sampling}^*$ is close to the global optimum f_{opt}^* .

5.7.3 Performance Investigation

In the following we present statistical performance results of our SDP subgraph matching approach for the more random subgraph matching problems discussed here that are created as described in section 5.7.1. In order to obtain reliable statistical results we have created 1000 problem instances for every different scene graph size. To compare these results with the results discussed the previous section we have computed the same performance measurements in this section as in section 5.6.4. As the exact global minima f_{opt}^* of the problem instances discussed here are computed by exhaustive search, we use the global optimum f_{opt}^* as reference point instead of the estimated optimum f_{est}^* used in

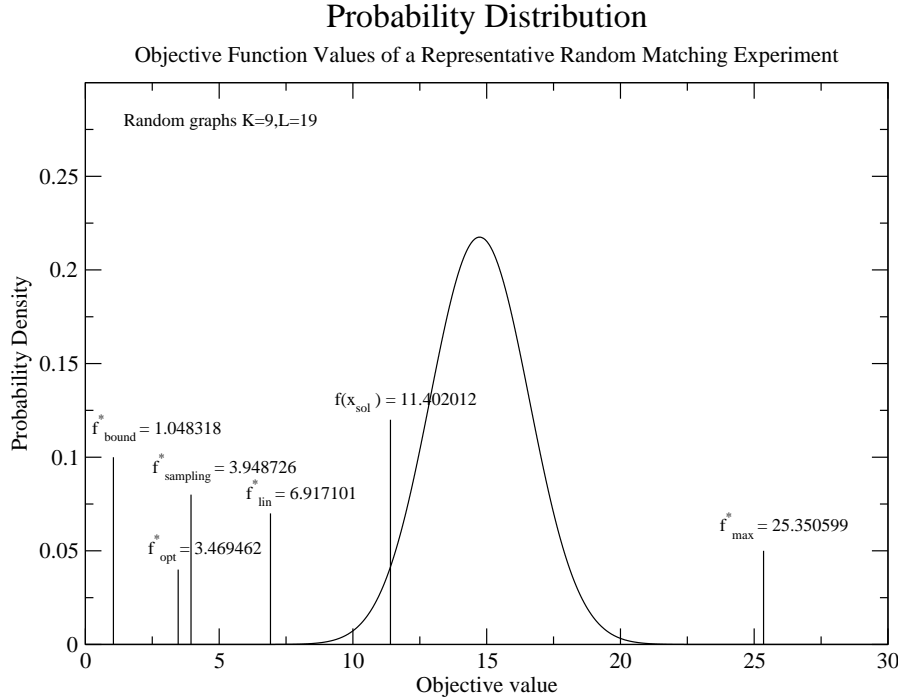


Figure 5.27: Probability distribution of objective values for a representative random graph matching problem with $K = 9$ and $L = 19$. The graph structure which belongs to this problem instance is shown in figure 5.26. For the here discussed subgraph matching problem instances the lower bound f_{bound}^* is usually not close to the global optimum f_{opt}^* which indicates that the relaxation is not tight.

the previous section.

Tightness

We investigate the quality of the lower bound f_{bound}^* obtained by the SDP relaxation (5.8) using the tightness measure $T(f_{\text{opt}}^*, f_{\text{bound}}^*) = (f_{\text{opt}}^* - f_{\text{bound}}^*) / f_{\text{opt}}^*$. Statistical results for this measure which compares the lower bound with the global optimum of (5.3) are shown in figure 5.28. Compared to the results shown in figure 5.21 we find that the lower bounds for the problem instances discussed here are usually not near the global optimum. This shows that the subgraph matching problem instances we are concerned with in this section are more difficult to be approximated by the SDP relaxation. In the following this is confirmed by the other performance measures.

Fraction of Optimally Solved Problem Instances

The fraction of optimally solved problem instances for the here discussed subgraph matching problems is depicted in figure 5.29. Compared to over 80%, the basic SDP subgraph matching approach results now only in less than 5% of all instances in the global optimum. The sampling still improves this rate to

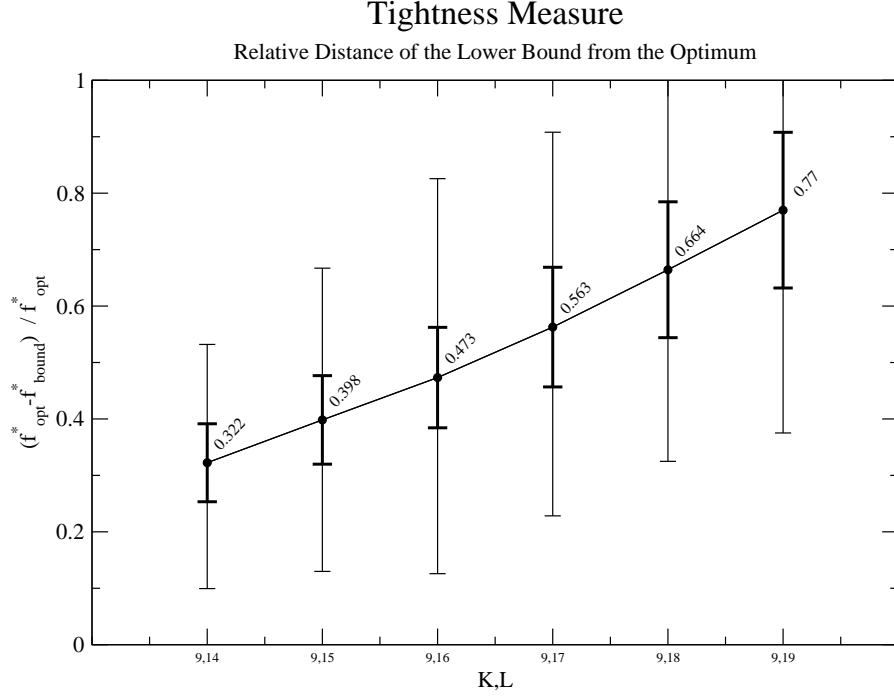


Figure 5.28: The tightness measure $T(f_{opt}^*, f_{bound}^*) = (f_{opt}^* - f_{bound}^*) / f_{opt}^*$ shows that for the here discussed more randomly created problem instances usually the lower bound is not near to the global optimum. For each problem size 1000 experiments were performed.

about 34% to 23% optimally solved problems but compared to the fraction of more than 96% for same sized instances discussed in the previous section (see figure 5.22) we observe a decreased ability to compute the global optimum. This supports the assumption that the performance of our subgraph matching approach depends much on the kind of problem data. The more structured problems discussed in the previous section are somehow easier to solve by the SDP relaxation.

Relative Optimum

To investigate the combinatorial solutions obtained by the SDP subgraph matching the mean values of the ratios f_{lin}^* / f_{opt}^* and $f_{sampling}^* / f_{opt}^*$ are shown in figure 5.30. We observe that the combinatorial solutions f_{lin}^* obtained by the basic SDP subgraph matching approach are generally not as close to the optimal objective value f_{opt}^* as we have observed for the problem instances discussed in the previous section (cf. figure 5.23).

The objective value f_{lin}^* is in the mean nearly 1.50 times larger than the global minimum f_{opt}^* for the problems of smaller size ($K = 9, L = 14$). This increases to a factor of nearly 2.0 for problem instances with $K = 9, L = 19$. For the previously discussed problem instances of the same size we have obtained ratio values close to 1.0 which means that these solutions are close to the optimum.

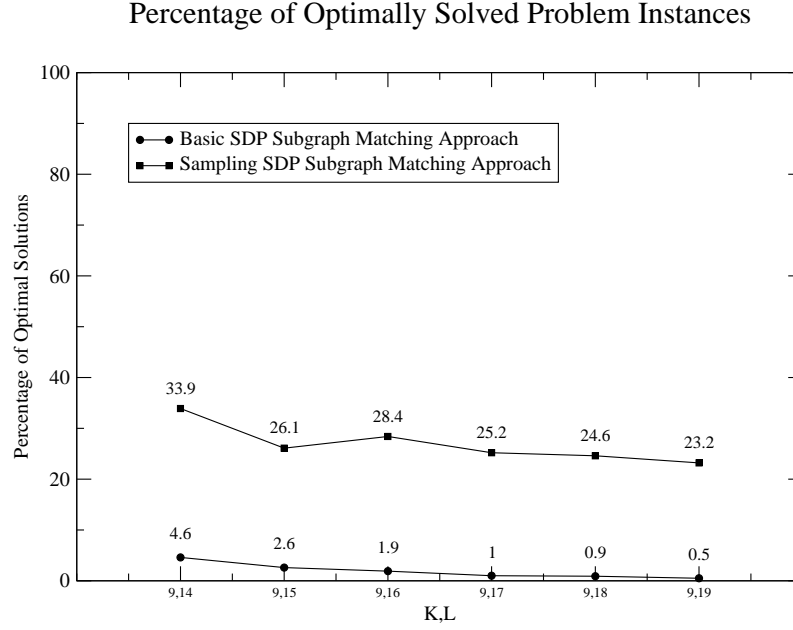


Figure 5.29: Fraction of optimal solved problem instances depending on the problem size. The sampling post-process is able to increase this fraction by exploiting the information in the non-integer solution. The results are based on 1000 problem instances for every problem size.

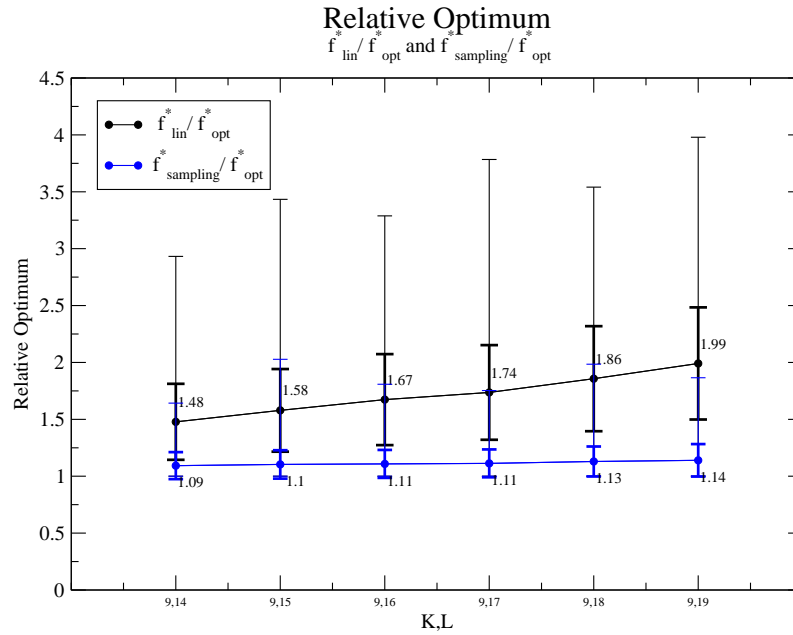


Figure 5.30: Mean values of the relative optima f_{lin}^*/f_{opt}^* and $f_{sampling}^*/f_{opt}^*$ for problem instances where the object and scene graphs are created randomly and independently. Besides the mean values the standard deviation and the corresponding minimal and maximal values are depicted. A ratio value of 1.0 indicates that the optimum value is reached.

Nevertheless, the sampling post-process was able to improve the combinatorial solution remarkably. But in the mean $f_{sampling}^*$ is still 1.09 to 1.14 times larger than the global optimum f_{opt}^* . For same sized problems discussed in the previous section (see figure 5.24) the sampling results usually in the global optimum.

This indicates again that the here investigated problem instances are harder to solve.

Optimal Mappings

In this part we investigate analogously to the previous section the number of mappings which are in accordance with the optimal mappings. For the problem instances we are concerned with here we define the 9 optimal mappings which belong to the optimal objective value f_{opt}^* to be also the desired mappings. The results for increasing scene graph size ($L = 14 - 19$) are depicted in figure 5.31. For the smaller experiments ($K = 9, L = 14$) one can expect that about 5 out

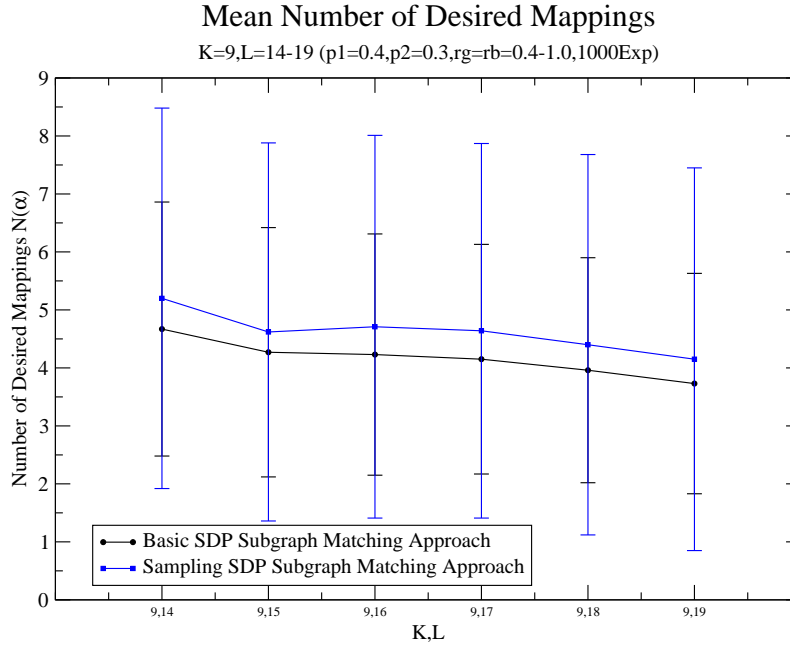


Figure 5.31: Mean number of mappings that are in accordance with the 9 optimal mappings. The number of correctly determined mappings obtained by the sampling SDP subgraph matching approach lies between 5.5 and 4 (out of 9) for the here discussed problems.

of the 9 obtained mappings are correctly detected. For problem instances with $K = 9$ and $L = 19$ this rate decreases to about 4 correctly detected mappings. This is in contrast to the expected number of correct obtained mappings for the problem instances discussed in the previous section. There we see in figure 5.25 that for problems of the same size nearly all mappings are in accordance with the desired mapping.

Summary

The comparison of the results in this section with the results in the previous section (section 5.6) reveal that beside the size of the problem instances also the kind of subgraph matching problem instances influences the performance of the SDP subgraph matching approach. The computed performance measures show that the more structured problem instances discussed in the previous section are better approximated by the SDP relaxation as the more random subgraph matching problems discussed in this section.

After this statistical performance investigation of our SDP subgraph matching approach we will show some real world subgraph matching problems in the next section.

5.8 Real World Examples

The aim of this section is to visualize the application of the SDP subgraph matching approach to real world problems. We first discuss in section 5.8.1 how we created our real world experiments and present in section 5.8.2 some results. Note that these examples have mainly the purpose to illustrate our SDP subgraph matching approach as we did not investigate in this thesis how reliable object and scene graphs can be obtained from images.

5.8.1 Creation of the Real World Examples

We have used a simple way to create the object and scene graphs from images for the here discussed subgraph matching problem instances. First, a corner detector [63] was used to extract feature points in a predefined region of an image. Selected feature points are supposed to represent the nodes of the graphs. The relational structure of the object and scene graph are created by a Delaunay triangulation [27, 129]. As the Delaunay triangulation is not always able to create graph structures in a stable manner we are sometimes obliged to create the graph structure manually. An example for node configurations where the triangulation is sensitive to perturbations of the feature points is shown in figure 5.32. Nevertheless, the similarity between the object and scene graph nodes

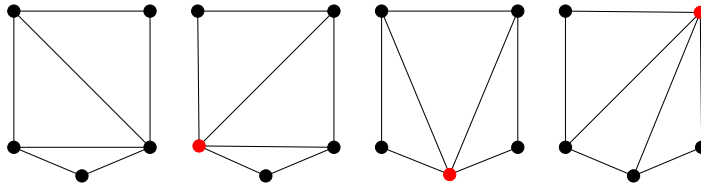


Figure 5.32: Example for similar node configurations that result in different Delaunay triangulations. The red marked nodes in the last three graphs are slightly moved toward the center of the graphs and the depicted different triangulations are obtained.

was calculated by the *Earth Mover Distance* from gray value histograms of a

quadratic area around the nodes. The Earth Mover Distance [123] is a similarity measure for histograms and can be stated as a linear optimization problem. For details concerning the Earth Mover Distance we refer to the Appendix B.2.

5.8.2 Results

In our first example a “matchbox graph” has to be matched against a scene graph which includes the matchbox as subgraph. In the second example a part of a “building brick” has to be matched against a scene graph. This problem instance includes also a small structural perturbation. The third example represents also a “building brick” subgraph matching instance. For this particular problem instance the SDP relaxation is tight enough to result already in the combinatorial optimum.

Subgraph Matching Example: Matchbox

The graph structures along with the images of our first real world subgraph matching example are shown in figure 5.33. The small graph in the left image

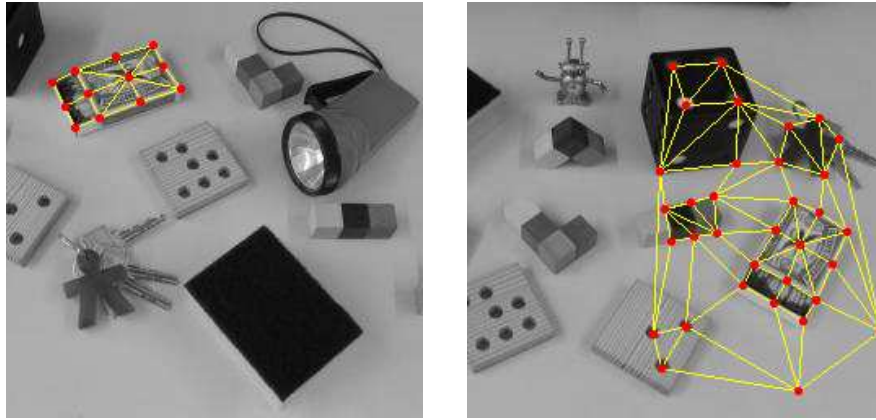


Figure 5.33: The “matchbox graph” in the left image with $K = 12$ nodes has to be matched against in the scene graph with $L = 34$ nodes in the right graph. The graph structures are created manually.

with $K = 12$ nodes is supposed to represent the object graph of the matchbox. The graph in the right image represents the scene graph with $L = 34$ nodes and contains the matchbox graph as a subgraph. In this case the graph structures were created manually. In figure 5.34 we show the matching that was obtained by the linear matching approach (5.2) which neglects every structural information. Only 4 of the 12 computed mappings are in accordance with the desired mapping. The correctly detected desired mappings are depicted by red lines while the undesired mappings are colored blue.

The SDP relaxation with $\alpha = 0.03$ results in the non-integer solution vector x_{sol} which is shown in figure 5.35. Interpreting the elements of x_{sol} as likelihood we see that the center node of the matchbox with the label 11 has a very high probability to be matched to the correct scene node with the label 15.

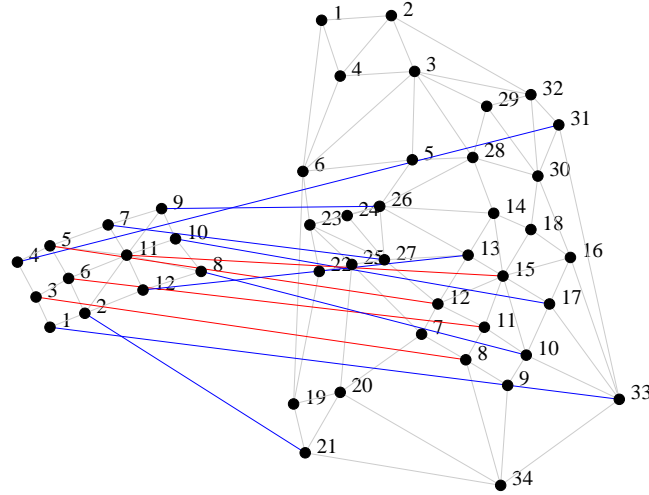


Figure 5.34: The shown mapping was obtained by the linear matching approach (5.2) which neglects every structural information of the graphs. Only 4 of the 12 mappings are in accordance with the desired mapping. The correctly detected desired mappings are depicted by red lines while the undesired mappings are colored blue.

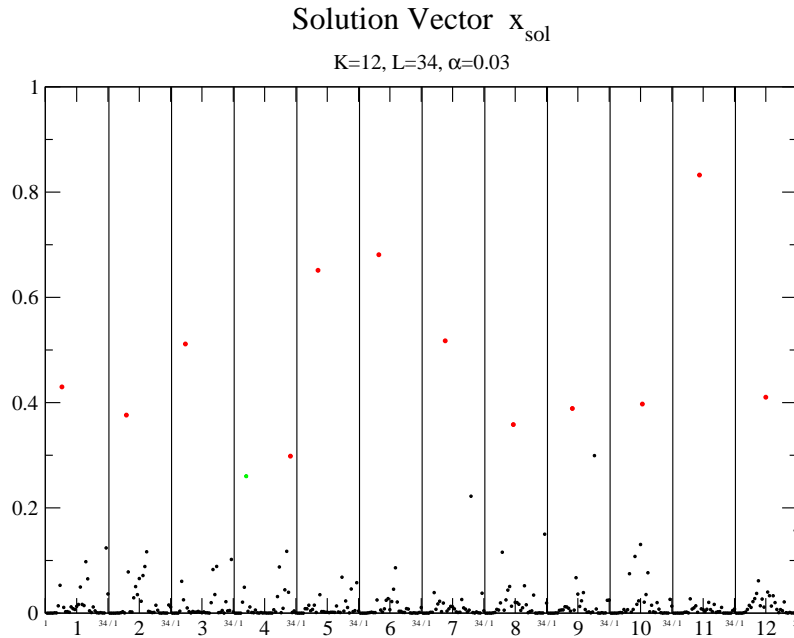


Figure 5.35: The non-integer solution vector x_{sol} obtained by the SDP relaxation with $\alpha = 0.03$. The linear post-process (5.23) selects the red colored elements as the likeliest mapping x_{lin}^* . The appendant matching is depicted in figure 5.36. The only undesired mapping $4 \mapsto 31$ is likely corrected by the sampling post-process as the desired mapping $4 \mapsto 7$ (green) is the second likeliest mapping for the object node 4. The correct matching obtained by the sampling post-process is shown in figure 5.37.

The linear post-process (5.23) obtains the red colored elements as 1 in the indicator vector $x_{lin}^* \in \{0,1\}^{KL}$. The other elements are 0. The appropriate matching is shown in figure 5.36. Only the mapping $4 \mapsto 31$ is an undesired

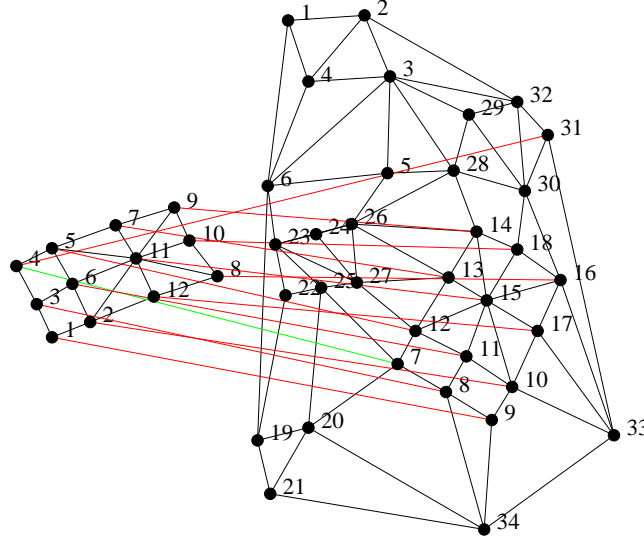


Figure 5.36: The basic SDP subgraph matching approach ($\alpha = 0.03$) results in the shown matching. Only the mapping $4 \mapsto 31$ is an undesired mapping. The green marked mapping $4 \mapsto 7$ is according to the solution vector x_{sol} shown in figure 5.35 the next best mapping for the object node with the label 4. Therefore it is very likely that the full desired mapping is obtained by the sampling post-processing.

mapping, but the next likeliest mapping $4 \mapsto 7$ which is shown as green line segment is the desired mapping. Therefore it is very likely that the full desired mapping is obtained by the sampling post-processing. We plotted the full desired mapping in figure 5.37.

We consider this subgraph matching example also to illustrate the changes in the solution vector x_{sol} if the parameter α was set too high. Therefore, we show in figure 5.38 the solution vector x_{sol} obtained by the SDP relaxation for $\alpha = 0.2$. One can see that x_{sol} reflects the preferred mappings less clearly and all the probabilities (except for the mapping $11 \mapsto 15$) are less than 0.2. This makes it harder for the sampling post-processing step to result in the global optimum. For the discussed problem instance the likeliest mapping obtained by the linear program (5.23) results in the mapping which is shown in figure 5.39. We observe that this is still close to the desired mapping but instead of one, now there are two undesired mappings: $2 \mapsto 20$ and $4 \mapsto 31$. Note that even in this case the $10 \cdot KL$ sampling post-processing steps result usually in the full desired matching which was already shown in figure 5.37. This shows once more that our approach is not too sensitive to the choice of the parameter α and that despite the bad choice for α a very good combinatorial solution was obtained.

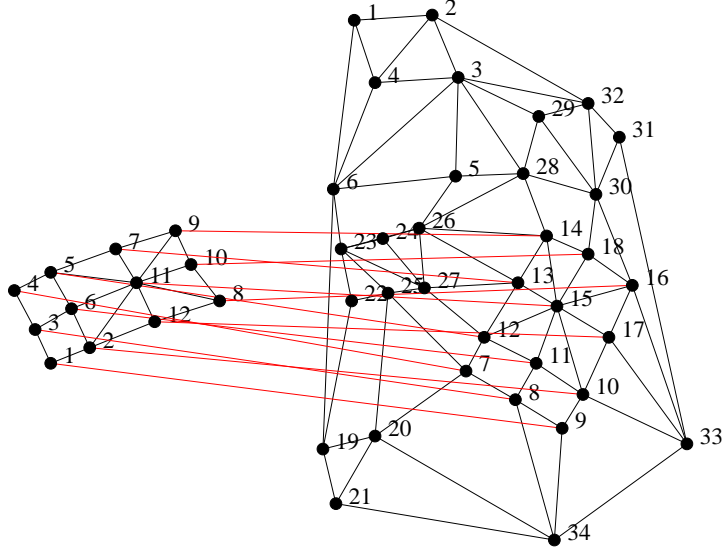


Figure 5.37: The full desired matching for the “matchbox” subgraph matching problem instance is obtained by the sampling SDP subgraph matching approach.

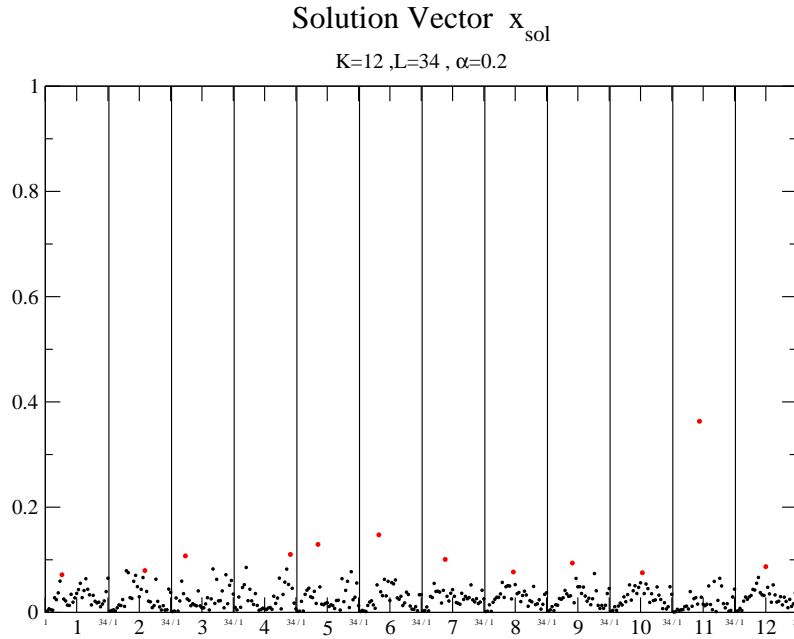


Figure 5.38: The solution vector x_{sol} obtained by the SDP relaxation with $\alpha = 0.20$. The solution becomes with increasing α less clear but many desired mappings are still the likeliest. The decreased tightness makes it harder for the sampling post-process to result in the global minimum.

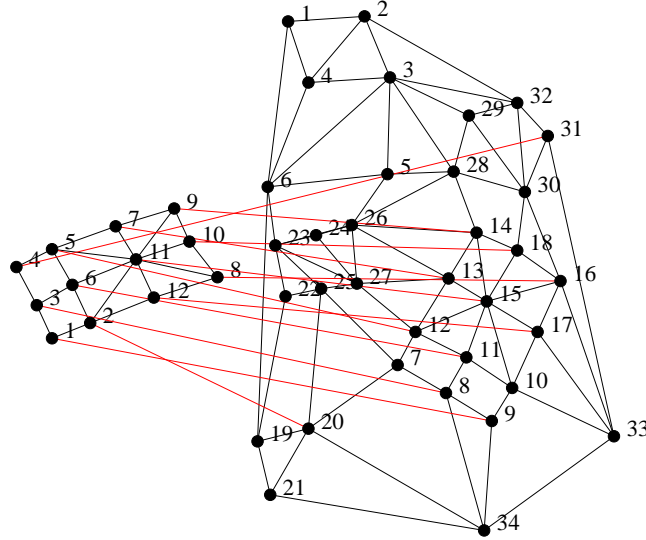


Figure 5.39: The matching was obtained by the basic SDP subgraph matching approach with $\alpha = 0.2$. Despite the bad choice of α , only the two mappings $2 \mapsto 20$ and $4 \mapsto 31$ are undesired mappings. Note that the sampling post-processing results usually in the full desired matching. This shows that the sampling SDP subgraph matching approach is not too sensitive to the choice of the parameter α .

Subgraph Matching Example: Building Brick 1

The second real world subgraph matching problem is shown in figure 5.40. The

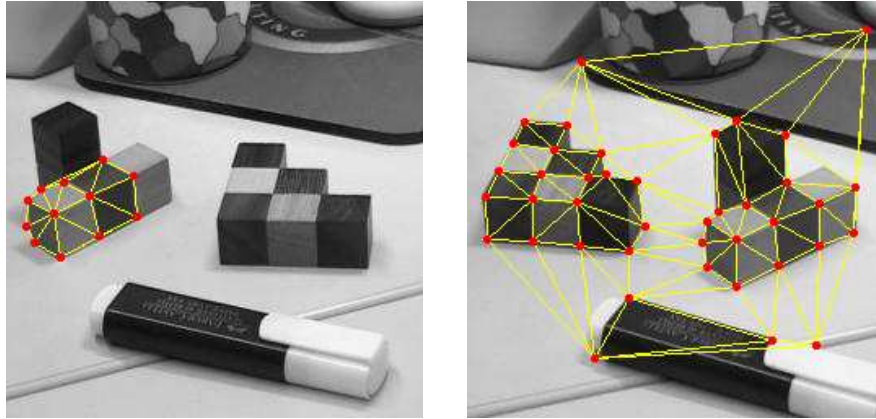


Figure 5.40: The object graph (left) with $K = 12$ nodes represents a part of a building brick and has to be matched against the scene graph (left) with $L = 41$ nodes. The graph structures are the result of a Delaunay triangulation of the feature points.

object graph with $K = 12$ nodes represents a part of a building brick and has to be matched against the scene graph with $L = 41$ nodes. This time the

graph structures are the result of a Delaunay triangulation of the shown feature points. Ignoring the structure, the linear matching approach (5.2) results in the undesired matching which is shown in figure 5.41. Only 3 of the 12 mappings

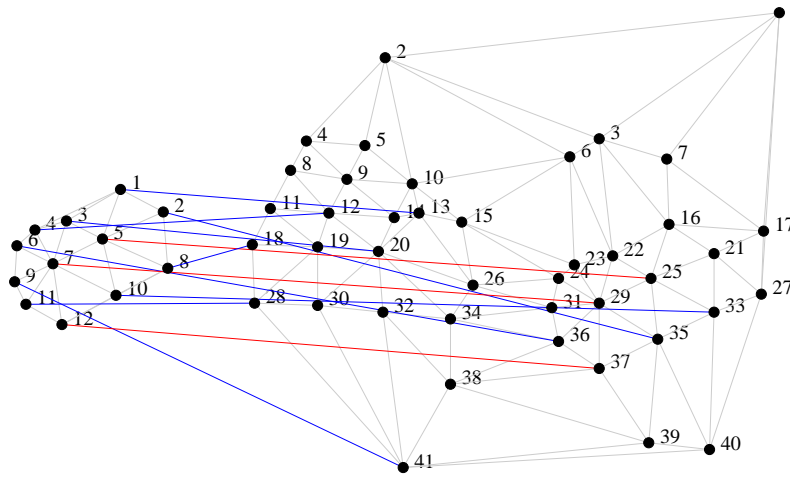


Figure 5.41: Ignoring the structure the linear program (5.2) results in the shown matching. Only 3 of the 12 mappings are in accordance with the desired mapping. The correctly detected desired mappings are depicted by red lines while the undesired mappings are colored blue.

are in accordance with the desired mappings. Many object nodes are mapped to scene nodes which belong to the second building brick as it has locally very similar features.

Taking a closer look at the graph structures one can observe that this example has a small structural perturbation. There is an additional edge between the nodes with the label “1” and “4” in the object graph which is not present in the scene graph.

The solution vector x_{sol} obtained by the SDP relaxation with $\alpha = 0.05$ is shown in figure 5.42. Even for this larger sized problem instance only a small fraction of the possible mappings have significantly increased probabilities in x_{sol} . This time already the vector x_{lin}^* computed by the basic SDP subgraph matching approach represents the desired mapping and the sampling post-processing step can not improve this solution any further. The appropriate mapping is shown in figure 5.43. Accordingly to the colors in the solution vector x_{sol} shown in figure 5.42, the second best mappings are shown by green line segments. One can observe that often the second best mappings are in accordance with the mappings computed by the linear matching approach and are therefore often seen as good mapping candidates.

We have seen that for the discussed problem instance the desired mapping was computed by the SDP subgraph matching approach and that the small structural error is no problem for the approach. We will discuss the issue of structural differences in more detail in section 5.9.3.

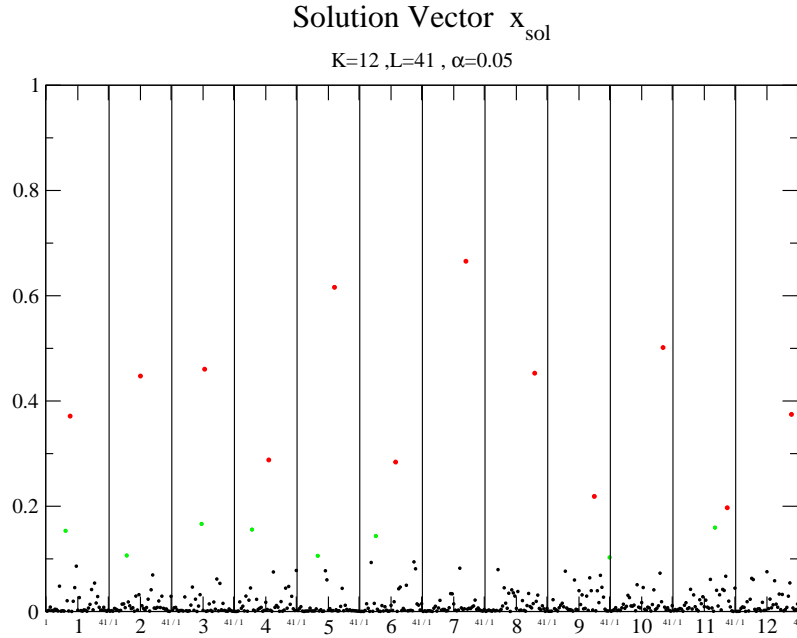


Figure 5.42: The non-integer solution x_{sol} obtained by the SDP relaxation with $\alpha = 0.05$ shows that only some mappings have an increased probability. In this case the likeliest mapping represents already the desired mapping and the sampling post-processing step is not required.

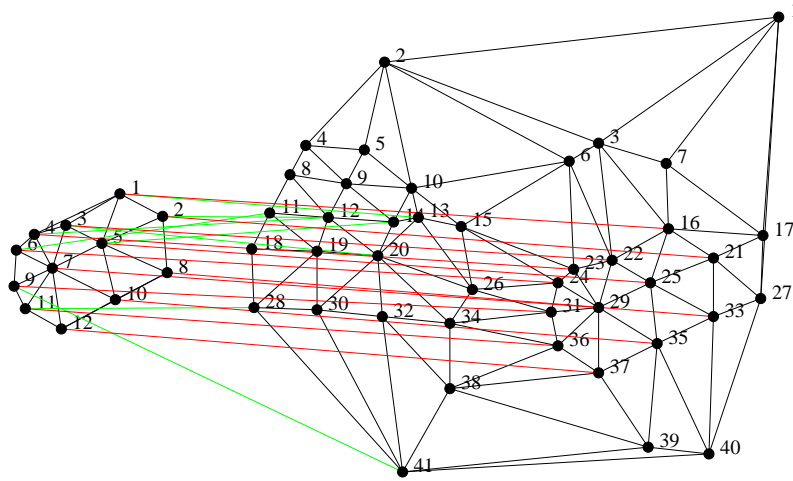


Figure 5.43: The basic SDP subgraph matching approach ($\alpha = 0.05$) results already in the desired mapping which is depicted by the red line segments. Some next best mapping candidates are shown by the green line segments.

Subgraph Matching Example: Building Brick 2

The third subgraph matching problem is based on the same pictures as the previous example and is depicted in figure 5.44. This time the object graph is

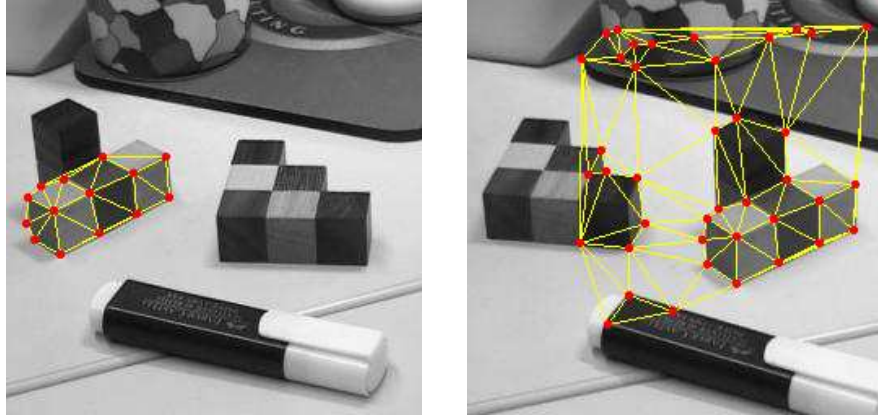


Figure 5.44: The part of the building brick with $K = 14$ nodes has to be matched against the scene graph with $L = 40$ nodes.

with $K = 14$ nodes slightly larger and a different scene graph with $L = 40$ nodes is chosen. The linear matching approach (5.2) results in the undesired matching which is shown in figure 5.45. Only 3 out of the 14 mappings are in accordance

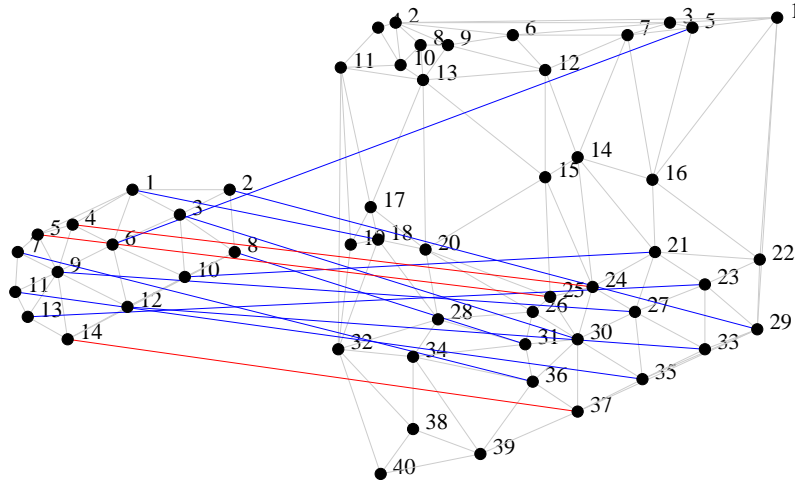


Figure 5.45: The linear solution for the real world graph matching problem shown in figure 5.44. Only three mappings ($4 \mapsto 24$, $5 \mapsto 25$, $14 \mapsto 37$) are in accordance with the desired mappings.

with the desired mappings. For this example we find that the SDP relaxation ($\alpha = 0.01$) is as tight as possible and already the solution vector x_{sol} results in a 0/1-integer solution. The computed solution vector x_{sol} is depicted in figure 5.46. As x_{sol} is also a feasible solution for the integer subgraph matching

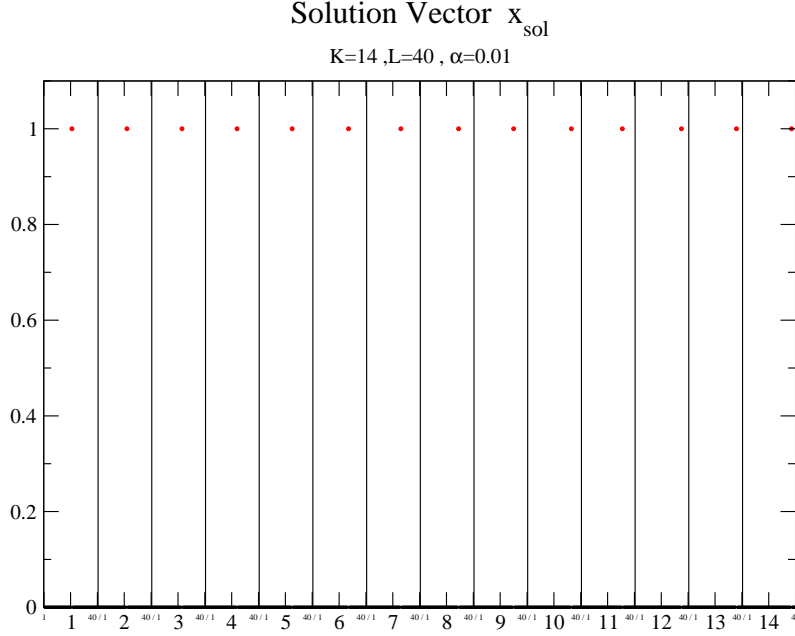


Figure 5.46: The solution x_{sol} of the SDP relaxation with $\alpha = 0.01$ results already in a 0/1-integer solution. Therefore this is guaranteed to be also the combinatorial optimum as the relaxed solution is a feasible solution for the combinatorial problem (5.3).

program (5.3) and the lower bound f_{bound}^* is equal to the combinatorial objective function $f(x_{sol})$ in (5.3) we conclude that x_{sol} must be the global optimum for (5.3) (see also section 3.1.1). The corresponding optimal mapping is shown in figure 5.47 and we see that it also represents the desired mapping. We observe that this particular problem can be approximated very good by the SDP relaxation. Although it is not clear to us we think a reason for this could be the fact that this problem instance has no ambiguous mappings which are cheap and simultaneously have a local fitting graph structure. We were astonished about this result especially as the example has the small structural perturbation which prohibited an isomorphic mapping of the object graph to the desired subgraph in the scene graph. Maybe this example can be used to find conditions under which a subgraph matching problem instance can be approximated as tight as possible by a convex relaxation.

5.9 Discussion

In this section we discuss several interesting aspects of the SDP subgraph matching approach. We first summarize the computational time and the memory requirements in section 5.9.1 and discuss in section 5.9.2 possible ways to reduce this computational effort. After that we investigate in section 5.9.3 the influence of structural perturbations on the performance of the subgraph matching approach. Then we discuss in section 5.9.4 the behavior of our approach in

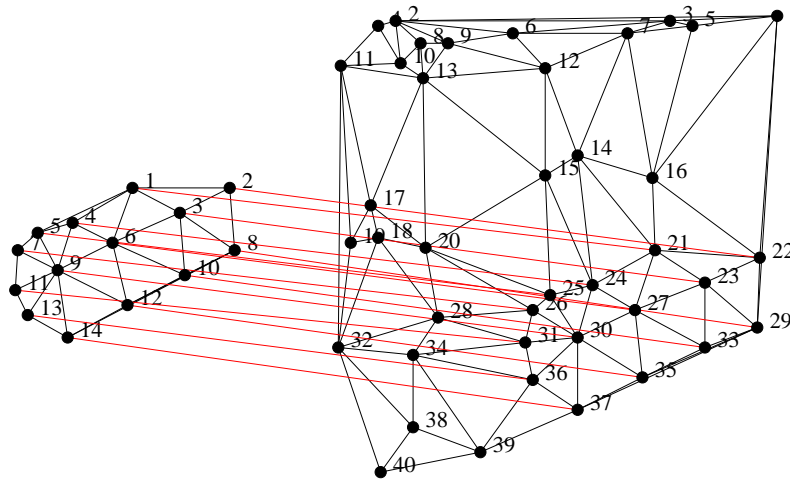


Figure 5.47: The optimal mapping and also the desired mapping was obtained by our SDP subgraph matching approach. For this example the solution of the SDP relaxation ($\alpha = 0.01$) coincides with the combinatorial optimum which makes any post-processing dispensable.

the case that the scene graph contains more than one potential good matching for the object graph. Thus we answer the question how bi- or multi-modal optimization problems are reflected in the convex relaxation which is by definition only unimodal. In section 5.9.5 we demonstrate that a simplification of our approach can be used to calculate a bound which sometimes proves that a subgraph isomorphism can not occur in a subgraph matching problem instance.

5.9.1 Computational Effort

The most computational effort within our SDP subgraph matching approach is needed for the computation of the solution of the SDP relaxation (5.8). We used external SDP solvers for this task and an independent benchmarking for several SDP solvers can be found in [104]. A comparison of three SDP solvers (DSDP [13], PENSDP [92] and CSDP [18]) on the basis of our own data is shown in table 5.1. There we compared the mean computation time needed to solve SDP

Problem Size (K/L)	CSDP 4.8	PENSDP 1.1	DSDP 4.7
9/14	(16±15)s	(42±11)s	(191±18)s
9/22	(89±46)s	(269±56)s	n.a.
9/30	(395±86)s	(1221±65)s	n.a.

Table 5.1: Mean computation time for three SDP solvers needed to solve SDP relaxations of several different sized subgraph matching experiments. The CSDP-solver fits best our needs.

relaxations of different sized subgraph matching problems which were created as described section 5.6.1. The solvers were run with default parameters on 3

GHz PCs with 2 GB storage. The computation time shows that the CSDP-solver fits best our needs. Note, that the DSDP-solver failed for the larger sized problems.

The CSDP-solver is a variant of the predictor corrector algorithm suggested by Helmberg, Rendl, Vanderbei, and Wolkowicz [70] and is described in detail in [18]. Nevertheless, below we sketch the storage requirement and the computational time for this solver.

Storage Requirement

In [19] the author points out that the CSDP-solver requires approximately

$$8(m^2 + 11(n_1^2 + n_2^2 + \cdots + n_s^2))$$

bytes of storage. Here m is the number of constraints and $n_1^2, n_2^2, \dots, n_s^2$ are the sizes of block diagonal matrices used to describe the $n \times n$ problem and solution matrices of the semidefinite program. Recall that in terms of the graph sizes K and L we have

$$n = KL + 1$$

and

$$m = 1 + K + \frac{K^2L}{2} + \frac{KL^2}{2}$$

With this we compute that a small subgraph matching problem instance with $K = 9$ and $L = 14$ needs about 17 MB while a larger problem instance with $K = 9$ and $L = 30$ needs already 220 MB of storage. About 1.8 GB of storage is needed to solve the SDP relaxation of the third real world example with $K = 14$ and $L = 40$ which is therefore near to the maximal problem size we can fully compute on today's computers with 2 GB of storage.

Computational Time

According to [18] the most difficult part in an iteration of the CSDP-solver is the computation of the Cholesky factorization of a $m \times m$ matrix which requires a time proportional to $O(m^3)$. This result is based on the here fulfilled assumption that individual constraint matrices have $O(1)$ non-zero elements and that m is somewhat larger than n .

In figure 5.48 we plotted the mean computation time used by the CSDP-solver to solve the problems discussed in section 5.6.4 on a 3 GHz Pentium 4 PC with 2 GB storage.

We have seen in this section that the computational demand increases quickly with the problem size. But one has to keep in mind that the original computational demand growth exponentially. And compared to this a single digit polynomial growth for a good approximation algorithm is already a big advantage. However, in the following section we discuss how we can reduce the size of the subgraph matching instances in order to reduce the computation time and to increase the size of problem instances we can cope with.

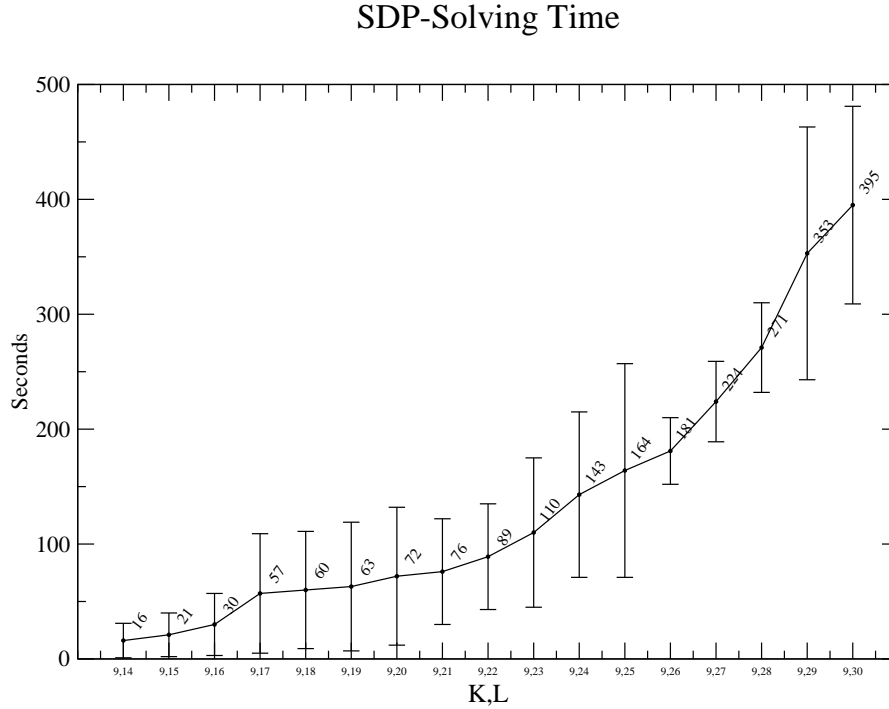


Figure 5.48: Mean time used by the CSDP-solver [18] on a 3 GHz Pentium 4 PC with 2 GB storage to solve the SDP relaxation for increasing subgraph matching problem size. The data is based on problem instances discussed in section 5.6.4.

5.9.2 Reducing the Computational Effort

The quick growth in the computational demand of the SDP relaxation restricts the application of our unmodified approach to small and medium sized problem instances and requires that larger problems must be somehow reduced in size. In the following we suggest two different approaches to reduce the computational effort. The first approach reduces the number of constraints for the problem instances by combining several constraints into a single constraint. The second approach reduces the problem size by eliminating undesired mappings directly from the problem instance.

Constraint Grouping

An easy method to reduce the number of constraints is to merge a predefined number of gangster constraints into a single constraint. Recall that the $(KK - K)L/2 + (LL - L)K/2$ gangster constraints represent the main fraction of the constraints in the SDP relaxation (5.8). Therefore, when reducing the number of these constraints we can expect a large computational gain. The gangster constraints (see also (5.12) and (5.13)) are defined by the non-zero elements of the matrix

$$G = I_K \otimes (E_{LL} - I_L) + (E_{KK} - I_K) \otimes I_L.$$

If we define M as the set of indices i and j where G has non-zero elements

$$M = \{(i, j) | i \leq j, G_{ij} = 1\}$$

the original gangster constraints are

$$\text{Tr}[A^{ij}X] = X_{ij} + X_{ji} = 2X_{ij} = 2X_{ji} = 0 \quad \forall (i, j) \in M$$

where one index pair (i, j) defines the position of symmetric elements in X which are constrained to be 0. Hence, the constraint matrix A^{ij} is a symmetric matrix which has only two elements set to 1 (see also section 5.4.2).

We can merge together two different gangster constraints with indices $(k, l) \neq (m, n)$ by adding their constraint matrices A^{kl} and A^{mn} :

$$\text{grouped2} A = A^{kl} + A^{mn}$$

The new constraint reads then:

$$\text{Tr}[\text{grouped2} AX] = X_{kl} + X_{lk} + X_{mn} + X_{nm} = 2X_{kl} + 2X_{mn} = 0.$$

We can see that this approach results in a less tight relaxation as the degree of freedom for the concerned elements in X increases. The zeros in X are no longer assured by the grouped constraints.

To group more gangster constraints together one can just add the appropriate number of constraint matrices to obtain a single constraint matrix. We note that we simply merged the gangster constraints in the order of occurrence in the matrix G , but other grouping strategies may result in an improved tightness of the relaxation.

The tightness measure $T(f_{est}^*, f_{bound}^*) = (f_{est}^* - f_{bound}^*)/f_{est}^*$ is plotted in figure 5.49 for 1000 subgraph matching instances of the size $K = 9$ and $L = 25$ which are created as discussed in section 5.6.1. There the constraints are reduced by successively merging 2, 4, 6, 8, 10 and 12 gangster constraints into a single constraint. We see that the tightness measure increases strongly with more constraints merged together. This means that the tightness itself decreases and we expect a less accurate approximation of the combinatorial subgraph matching approach.

The influence of the capability to compute the global optimum is shown in figure 5.50. If more than two constraints are merged together the basic SDP subgraph matching approach is only in less than 5% of the instances able to compute the global optimum. But the sampling increases this rate and even for 12 grouped gangster constraints a rate larger than 35% is reached.

The advantage of this approach is that with a decreasing number of constraints the time needed to solve the SDP relaxation decreases largely. But one has to weigh up the time benefit to the loss of tightness. In figure 5.51 the mean time needed by the CSDP-solver to solve the SDP relaxations is plotted. The time was measured on a 3 GHz Pentium 4 PC with 2 GB storage. Merging 2 gangster constraints into a single constraint reduces the mean computation time by a factor of about 5.

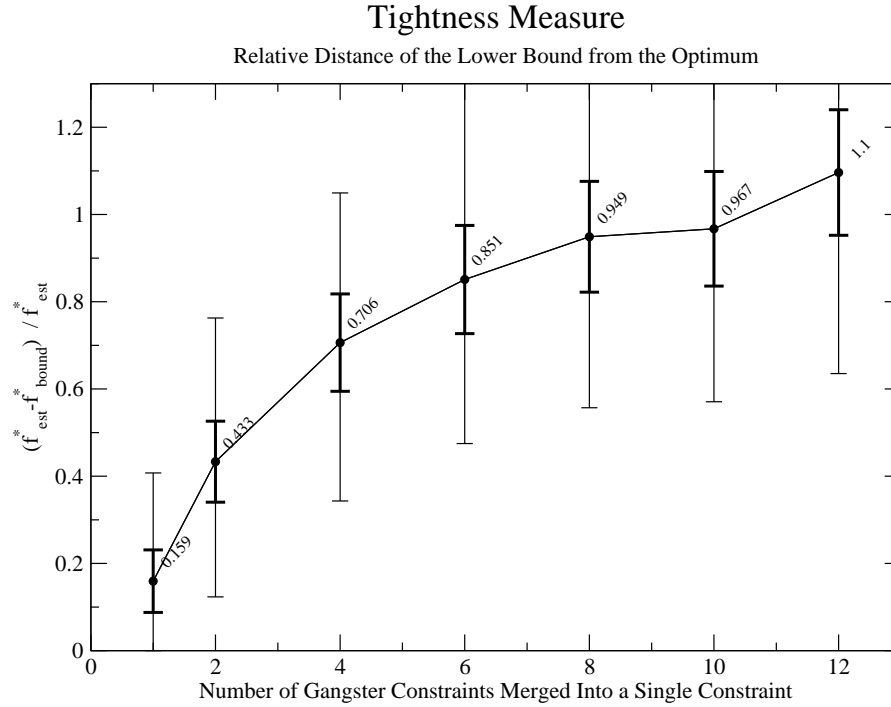


Figure 5.49: The grouping of gangster constraints reduces the tightness of the SDP relaxation which means that our tightness measure $T(f_{est}^*, f_{bound}^*)$ increases. The results are based on 1000 subgraph matching problem instances with $K = 9$ and $L = 25$ which are created as discussed in section 5.6.1

Problem	orig.	2	6	12
Matchbox	3489s (9397)	330s (4909)	49s (1917)	33s (1169)
Building Brick 1	5505s (13051)	783s (6778)	96s (2596)	65s (1551)
Building Brick 2	8243s (15135)	1443s (7855)	164s (3002)	101s (1789)

Table 5.2: The time used to solve the SDP relaxation decreases drastically when 2,6 or 12 gangster constraints are merged into a single constraint. The total number of constraints in the relaxation is plotted in brackets.

The impact of this approach to our real world examples is shown in the tables 5.2 and 5.3. In table 5.2 one can see that the time used to solve the SDP relaxation improves drastically when several gangster constraints were merged into a single constraint. But the quality of the solution decreases and in table 5.3 we see that the basic SDP subgraph matching approach computes more often undesired mappings when the tightness measure increases. Nevertheless, the sampling is usually able to correct these undesired mappings for our real world examples.

As time is often an important factor in an application one can benefit to merge two or eventually four constraints into a single constraint without losing too much accuracy. Next we discuss a way to reduce the problem size by removing

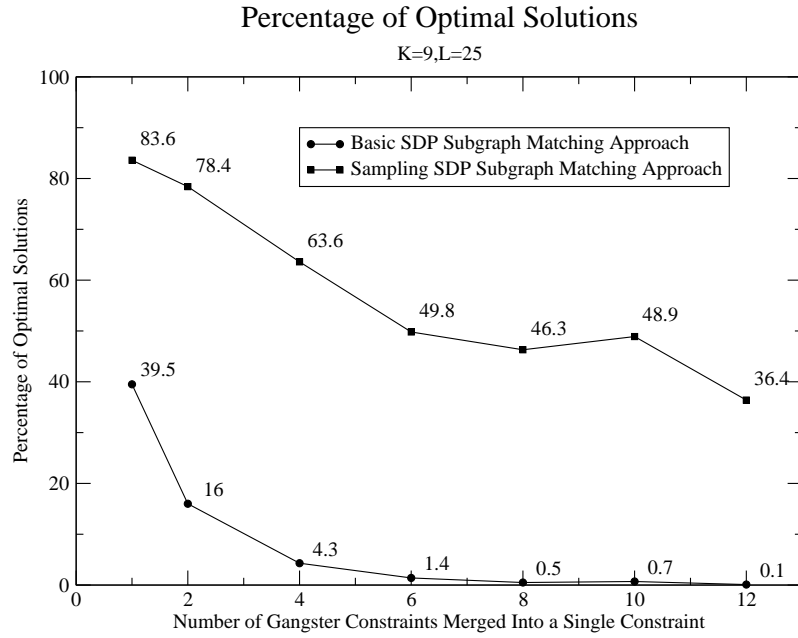


Figure 5.50: Percentage of optimal solved subgraph matching problems depending on the number of merged gangster constraints. The lesser tightness of the SDP relaxation leads to a strong decrease of the fraction of optimal solved problem instances.

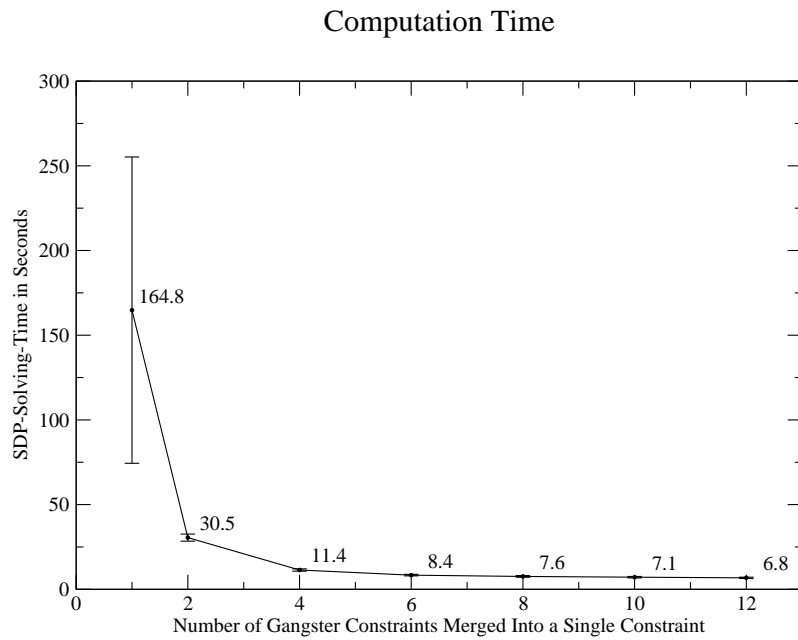


Figure 5.51: The mean time needed on a 3 GHz Pentium 4 PC for solving the SDP relaxation. The computation time decreases with a decreasing number of constraints. However, one has to weigh up the time benefit to the loss of tightness.

Problem (K,L)	orig.	2	6	12
Matchbox (K=12,L=34)	0.110 (1;0)	0.266 (3;0)	0.559 (5;0)	0.836 (8;2)
Building Brick 1 (K=12,L=41)	0.189 (0;0)	0.643 (2;0)	1.184 (5;0)	1.478 (9;0)
Building Brick 2 (K=14,L=40)	0.000 (0;0)	0.005 (0;0)	0.048 (0;0)	0.083 (2;0)

Table 5.3: The computed tightness measure $T(f_{est}^*, f_{bound}^*)$ increases when 2,6 and 12 gangster constraints are merged into a single constraint. The numbers of undesired mappings obtained by the basic and the sampling SDP subgraph matching approach respectively are shown in brackets. With decreased tightness (increased tightness measure T) the basic subgraph matching approach fails more likely. But the sampling is usually able to correct these undesired mappings.

mappings directly from the original subgraph matching problem.

Reducing the Dimension of the Problem

A mapping of the object node i to the scene graph node j is represented in the combinatorial subgraph matching approach (5.3) by the binary variable $x_{ji} = \{0, 1\}$ of the vector x . If this mapping is considered to be unlikely or undesired, it can be forced to be zero in the solution to indicate that this mapping does not occur. Forcing $x_{ji} = 0$ is equivalent to eliminating this particular mapping from the original combinatorial problem (5.3) and results in a problem formulation where the dimension of the vector x is reduced by one.

This reduction of the problem size can easily be implemented by removing the elements x_{ji} and w_{ji} from the vectors x and w and by removing the appropriate row and column from the quadratic problem matrix Q in the combinatorial problem formulation (5.3). Furthermore the constraints must be adapted by deleting the appropriate columns from the constraint matrices A_K and A_L . The SDP relaxation of this reduced problem is computed straight forward as it is a reduced version of the original relaxation.

This approach enables us to use prior knowledge about undesired mappings to eliminate them from the subgraph matching problem formulation. However, for general inexact subgraph matching problems it is difficult to find a strategy to eliminate mappings such that in fact only undesired mappings are removed. Of course, if this would not be hard this method would be able to solve the original problem.

For the sake of simplicity we have only applied a simple threshold τ to eliminate mappings $i \mapsto j$ if its similarity measure is above the threshold ($w_{ji} \geq \tau$). In figure 5.52 we show the solution vector x_{sol} for the “Building Brick 1” example where 292 undesired mappings of the 492 possible mappings were eliminated. These 292 elements are set to zero in the shown vector. We can see that this makes the solution vector x_{sol} more clearly (compare with figure 5.42) as the desired mappings become more likely. Furthermore the computation time for the solution of the SDP relaxation is reduced from 5505 to 44 seconds.

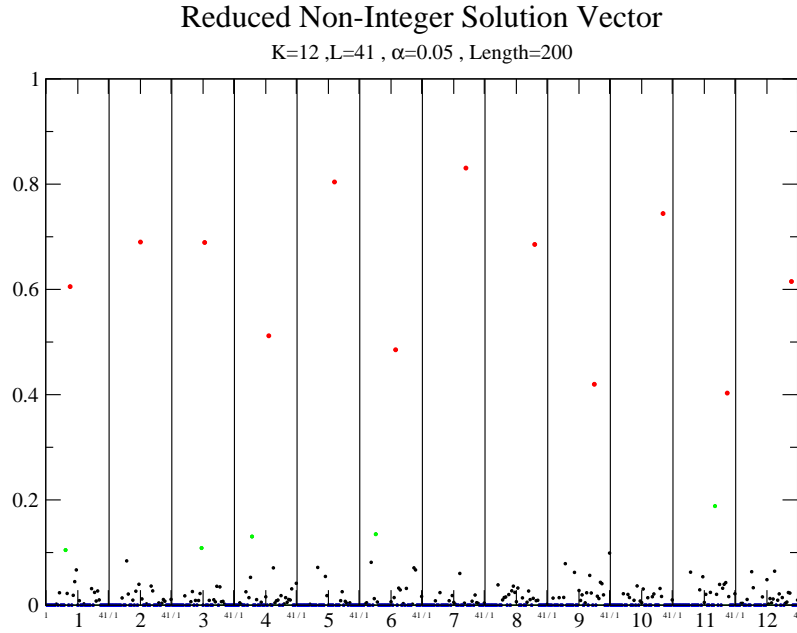


Figure 5.52: With a simple thresholding the problem size was reduced from $n = KL + 1 = 493$ to $n = 200$. The reduced solution x_{sol} has the same characteristics as the solution vector shown in figure 5.42.

In figure 5.53 a real world example is shown which is too large to be computed without any size reduction. The object graph has $K = 20$ and the scene graph

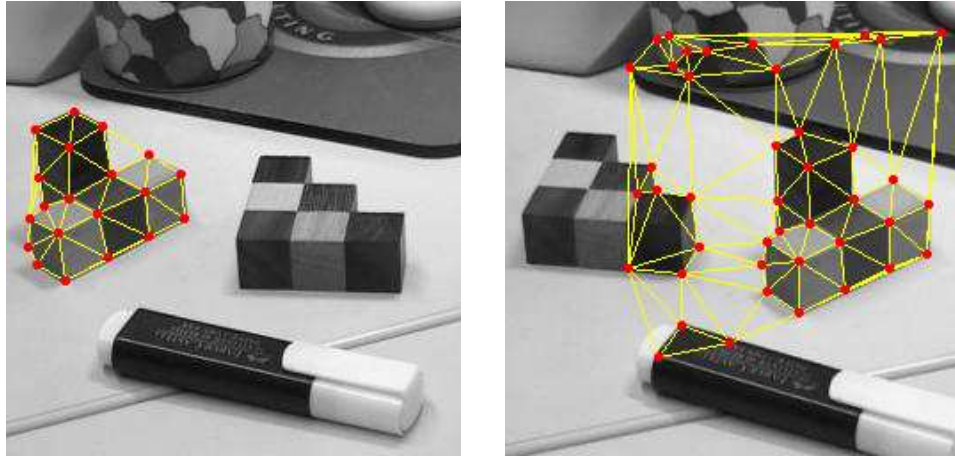


Figure 5.53: Examples for a real world example $K = 20, L = 43$; 573 edges of 860 selected

$L = 43$ nodes. For this problem size 5.7 GB of storage must be available. We reduced the 860 possible mappings to 573 possible mappings by eliminating the 287 most costly mappings. The vector x_{sol} for this reduced problem is shown in figure 5.54. The desired combinatorial solution is already obtained by the

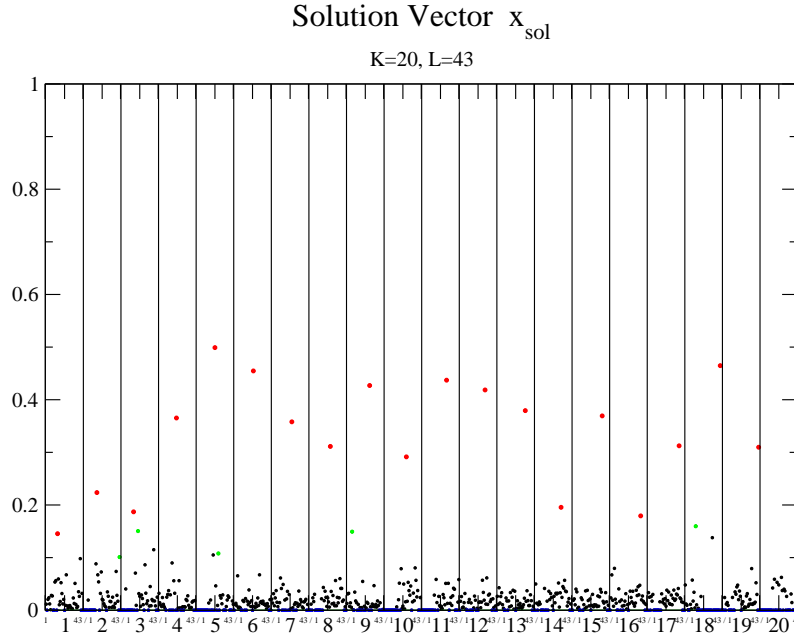


Figure 5.54: Solution vector x_{sol} obtained by the SDP relaxation of a size reduced subgraph matching problem shown in figure 5.53. The 860 possible mappings are reduced to 573 considered mappings by eliminating the 287 most costly mappings. The elements which belong to the eliminated mappings are set to zero.

linear post-processing (5.23) without the need of the sampling post-processing. The appropriate mapping is depicted in figure 5.55.

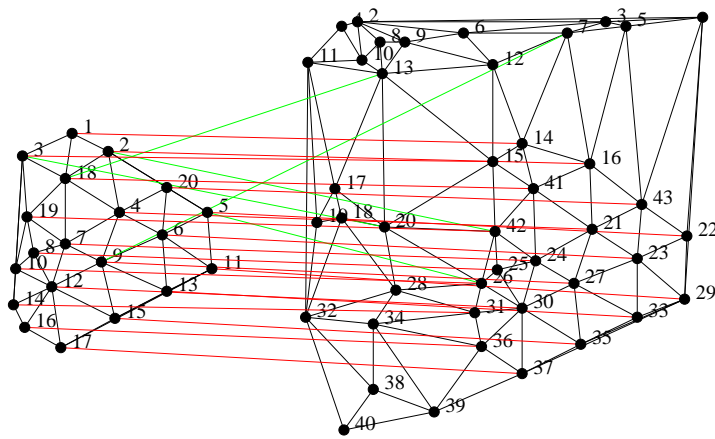


Figure 5.55: The full desired mapping which is shown by the red line segments is obtained by the basic SDP subgraph matching approach. Some next best mappings with a probability > 0.1 are depicted by green line segments. The problem size was reduced by eliminating the 287 most costly mappings from the 860 possible mappings of the original problem instance.

The elimination of undesired mappings from the subgraph matching problems lead to an improved solution x_{sol} such that the decision for a combinatorial solution can be made more clearly and reduces the computation time. Therefore, if one is able to determine undesired mappings a priori one should exploit these knowledge to eliminate these mappings from the original problem instance. Of course, if one is able to determine a desired mapping, this and all appropriate undesired mappings can be removed from the problem instance too.

The observation that the solution vector x_{sol} shows the preferred mappings more clearly for a reduced problem size brings up the idea for an algorithm that possibly computes a very good approximation to the combinatorial optimum. It makes use of the vector x_{sol} to determine the mappings that can be eliminated from the original problem. The idea is to eliminate successively the mappings from the subgraph matching instance which have the lowest probability according to the solution vector x_{sol} of the relaxation. The reduced problem is likely to result in a new solution vector x'_{sol} which shows more clearly the preferred mappings. This should be iterated until the problem size is small enough that the solution can be obtained fast by exhaustive search. We think this is a very promising algorithm to compute the combinatorial approximation and we believe that it is very likely that this approach results in the global combinatorial optimum because of the observation that optimal mappings are somehow preferred in x_{sol} and are therefore less likely removed from the original subgraph matching problem.

5.9.3 Structural Perturbations

In section 5.6 we have implicitly assumed that in our graph matching problem instances only perturbations occur which are related to the similarity measure w between the nodes of the object and scene graph. But, from the computer vision point of view it is also important to investigate the impact of structural perturbations which may occur when the graph matching problems are obtained by a vision system. To this end, we have investigated the mean number of computed mappings which are in accordance with the desired matching for an increased perturbation in the graph structure.

We have created 1000 problem instances with $K = 9$ and $L = 25$ which are created as described in section 5.6.1. Then we successively removed 3 edges in the scene graph. We note that the edges are always removed from the part of the scene graph which represents the object graph structure. In figure 5.56 an example for such a problem instance is shown. The 9 desired mappings are represented by red lines and 3 randomly selected edges, which are removed successively from the graph structure, are marked blue.

In figure 5.57 we have plotted the mean number of correctly detected desired mappings that are obtained by the SDP subgraph matching approach for the described experiments. One can see that an increased perturbation decreases the number of mappings we can expect to be in accordance with the desired mapping. Particularly, for our experiments we can expect to get 8 out of 9 desired mappings correctly if no structural error is present (cf. section 5.6). If three edges are removed we can expect to obtain only about 5 desired map-

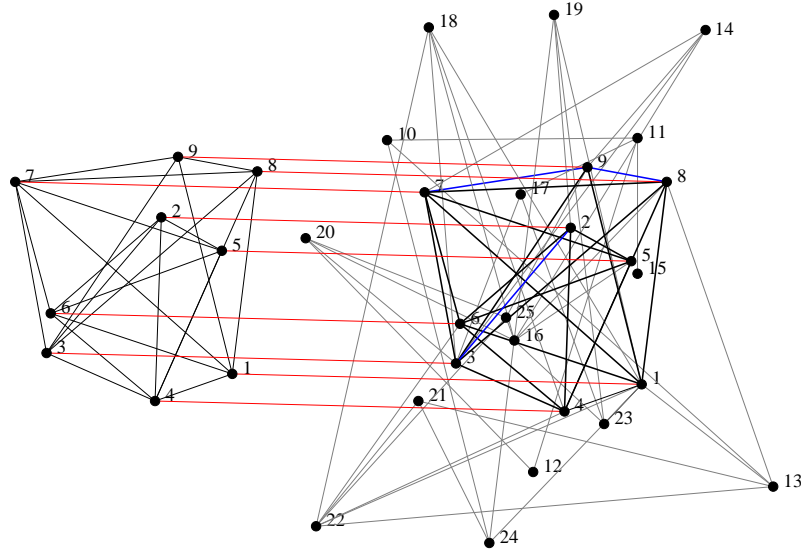


Figure 5.56: Example for a random subgraph matching problem with noise. The desired mappings are represented by red lines. In our experiments three randomly selected edges (here marked blue) are successively removed from the part of the scene graph which represents the object graph structure.

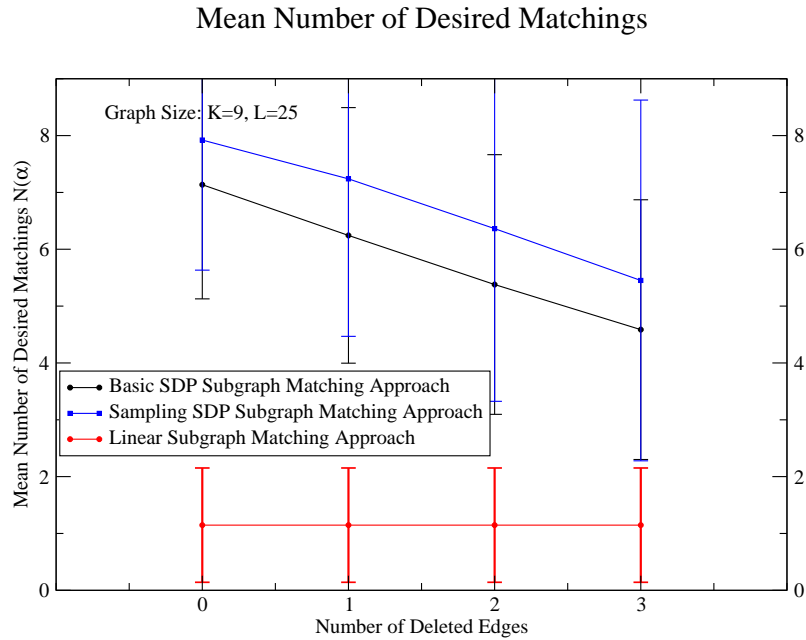


Figure 5.57: Mean number of mappings which are in accordance with the desired mapping for an increasing perturbation in the graph structure. The number of desired mappings decreases with increasing perturbation as a partly undesired mapping becomes more likely to represent the global minimum of (5.3). For the unperturbed problem instances we can expect 8 out of 9 mappings to be in accordance with the desired mapping which decreases to about 5 desired mappings for the problem instances where 3 edges are removed.

pings correctly. This shows that our approach is somehow sensitive to structural perturbations in the underlying graph structure. One reason for this behavior could be found in the modeling of the combinatorial subgraph matching approach (5.3). The objective value for the desired mapping is increased by 2α for each structural difference between the object graph structure and the mapped subgraph structure. Therefore, with increasing perturbation, it becomes more likely that a better global minimum, which our approach tries to find, results in a partly undesired mapping. However, the obtained results are likely to be improved if the reliability of the similarity measure between the nodes of the two graphs is improved.

5.9.4 Bimodal Experiments

In this section we will answer the question what happens if the scene graph contains more than one potential good matching for the object graph. This means that the combinatorial objective function of (5.3) has more than one very good local minima for different matchings. As the convex SDP relaxation has only one global minimum we are interested in how the convex approach reflects such a multi-modal situation.

To investigate this kind of problem we have modified the illustrative subgraph experiment discussed in section 5.5. We have copied the scene graph of this experiment twice to create an enlarged scene graph with $L = 26$ nodes. The object graph with $K = 5$ nodes was left unchanged. Furthermore, we have utilized the same similarity measure for both parts of the scene graph. This creates a symmetric problem instance where the global minimum of (5.3) is attained for the following two different desired matchings:

First desired matching: $(1 \mapsto 9, 2 \mapsto 1, 3 \mapsto 8, 4 \mapsto 4, 5 \mapsto 2)$

Second desired matching: $(1 \mapsto 22, 2 \mapsto 23, 3 \mapsto 24, 4 \mapsto 25, 5 \mapsto 26)$

We plotted the problem instance along with several possible mappings obtained by the SDP subgraph matching approach in figure 5.58. The parameter was set to $\alpha = 0.36$. The red lines represents the likeliest matching of the object graph to the right part of the scene graph. The green lines represents the likeliest matching of the object graph to the left part of the scene graph. We can see that only the mappings for the object node 4 are undesired mappings. However, the next best mapping candidates, depicted by the cyan and blue colored lines, are desired mappings for this node.

To understand the shown mappings in figure 5.59 the non-integer solution x_{sol} obtained by the SDP relaxation (5.8) is shown. Appropriate vector elements are colored in correspondence to the colors of the mappings that are shown in figure 5.58. We observe that the solution vector x_{sol} reflects the symmetry of this problem instance. Assume that j and k are “equivalent” nodes in the left and in the right part of the scene graph. Then we get for this problem instance the same solution values $(x_{sol})_{ji} = (x_{sol})_{ki}$ for the mappings of the object node i to the scene graph nodes j and k . Therefore, from the probabilistic point of view, the equivalent mappings have the same probability which meets our expecta-

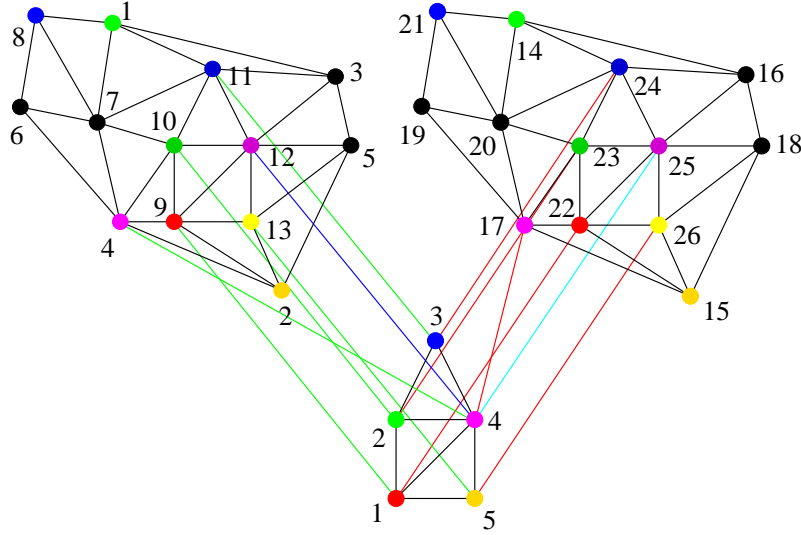


Figure 5.58: Bimodal subgraph problem instance which has two potential good matchings for the object graph $K = 5$ in the scene graph $L = 26$. The green lines represent the likeliest mappings of the object nodes to the left part of the scene graph while the red lines represent the likeliest mapping to the right part of the scene graph. The blue and cyan colored line represents the next best candidates for the mapping of object node 4 into the scene graph.

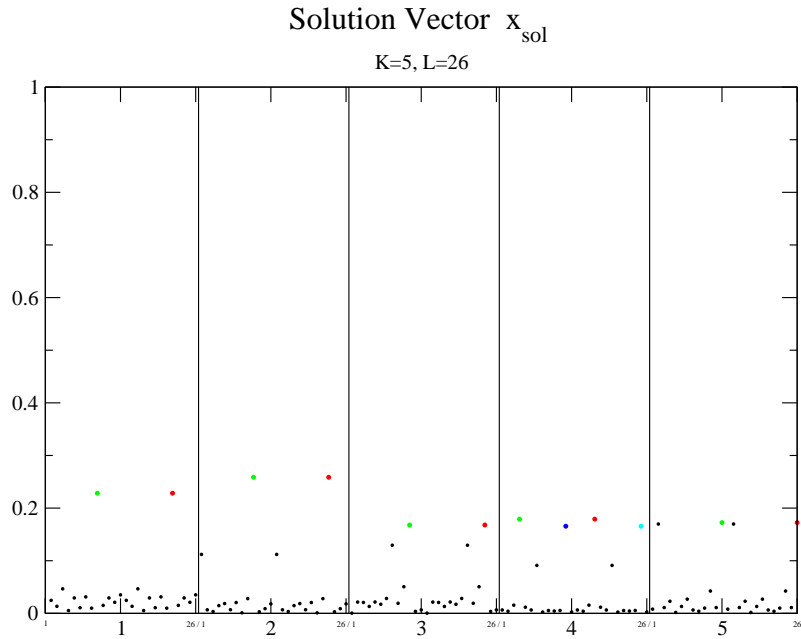


Figure 5.59: Non-integer solution vector for the bimodal experiment ($\alpha = 0.36$). The vector elements are colored consistent with the colors of the mappings which are shown in figure 5.58. All the desired mappings have high probabilities compared to the undesired mappings.

tion as the problem is symmetric. However, the important fact is that all the desired mappings have a high probability compared to other mappings. Note that this is also true for less symmetric situations. That makes it likely that the sampling post-process can extract a good combinatorial solution. Indeed, repeated experiments have shown that the sampling post-process results nearly always in one of the two desired mappings which are depicted in figure 5.60. This artificial bimodal experiment shows that the solution vector x_{sol} contains

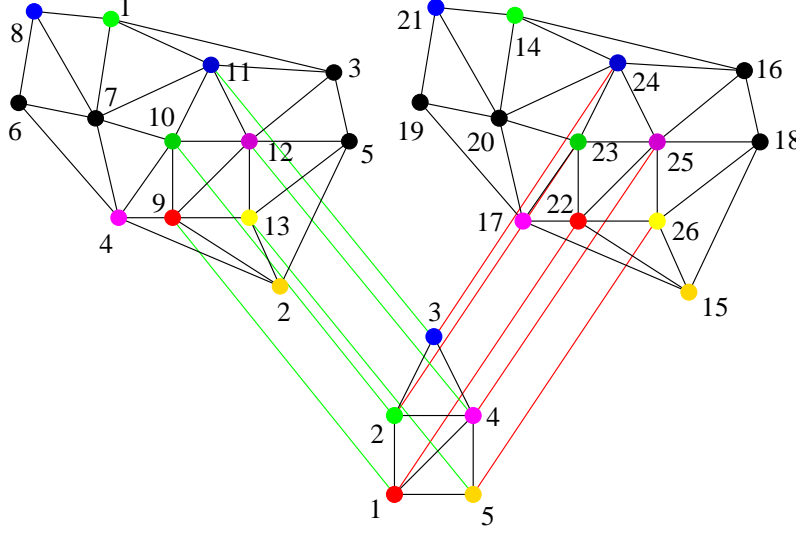


Figure 5.60: Either of the two desired matchings represented by the green and red line segments is very likely obtained by the SDP graph matching approach followed by the sampling post-process. The non-integer solution has a probabilistic “knowledge” of both desired matchings.

the “knowledge” of both desired matchings. This knowledge is represented by the higher probabilities for all the “good” mappings.

We expect a similar behavior for multi-modal experiments. But with an increasing number of objects that are present in the scene graph, the solution vector x_{sol} becomes more scattered and the increased ambiguity makes it harder for the sampling post-processing to result in a good and desired matching.

5.9.5 A New Bound for Subgraph Non-Isomorphism

In this section we suggest a new lower bound, computed by the SDP relaxation (5.8), that can sometimes be used to decide if a subgraph isomorphism cannot occur in graph matching problems (cf. section 5.3.2). Here we consider problem instances where only the structures of the object and scene graph are given and therefore we deal with simple graphs. The subgraph isomorphism problem can be stated as the combinatorial optimization problem (5.7). The idea is based on the fact that a subgraph isomorphism for such problem instances always leads to 0 as optimal objective value for (5.7). Therefore, a lower bound for (5.7) that is larger than 0 represents a proof that a subgraph isomorphism cannot occur

in the problem instance. Note that conversely, a negative lower bound does not imply that a subgraph isomorphism must occur and only indicates that a subgraph isomorphism is possible.

An example for such subgraph isomorphism problems is depicted in figure 5.61. Setting the similarity vector $w = 0$ the combinatorial subgraph isomorphism

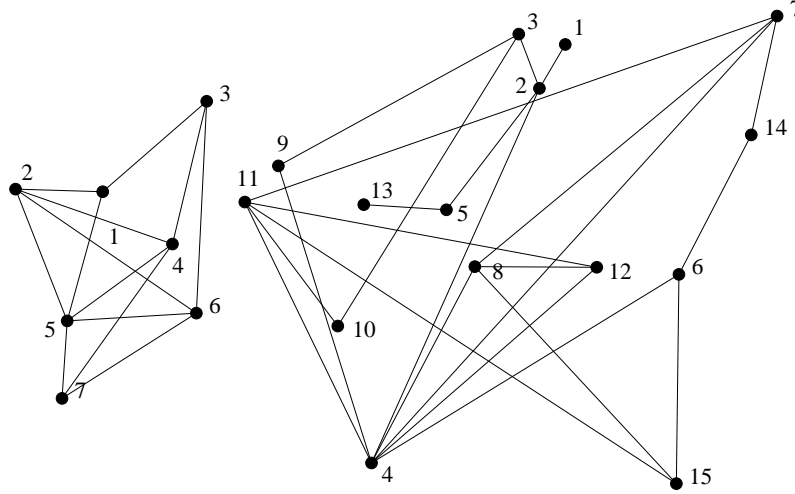


Figure 5.61: Example for a randomly created subgraph problem. Is there a subgraph isomorphism? For the shown problem instance we can compute a lower bound > 0 for (5.7) which proves that no subgraph isomorphism is present.

problem (5.7) is equal to our combinatorial subgraph matching approach (5.3). With that we can use the SDP relaxation of (5.3) to calculate a lower bound for (5.7). For the example shown in figure 5.61 we compute a lower bound > 0 using the SDP relaxation (5.8), which proves that a subgraph isomorphism does not exist in this problem instance. Note that we did not eliminate mappings that could not lead to an subgraph isomorphism.

The possible objective values of (5.3) are restricted to discrete values as the quadratic term $\alpha x^\top Q x$ can only reach values which are multiples of 2α . The parameter α is just a scaling parameter for this kind of problems and has no influence on the solution. Therefore it can be set to an arbitrary value $\alpha > 0$. The discrete distribution of the objective values for the subgraph isomorphism problem shown in figure 5.61 is depicted in figure 5.62 where we have set $\alpha = 0.3$.

For a first preliminary investigation of this bound we created 1000 small subgraph matching problem instances with a similarity vector $w = 0$. We have chosen the size of the object and scene graph to be $K = 7$ and $L = 15$, respectively. The edge probability of the object graph was set to 0.5 and the probability for an edge in the scene graph was set to 0.2.

The results for this experiment series reveal that for various problem instances it is indeed possible to conclude that no subgraph isomorphism exist. We have obtained 388 problem instances with a lower bound > 0.0 which proves that

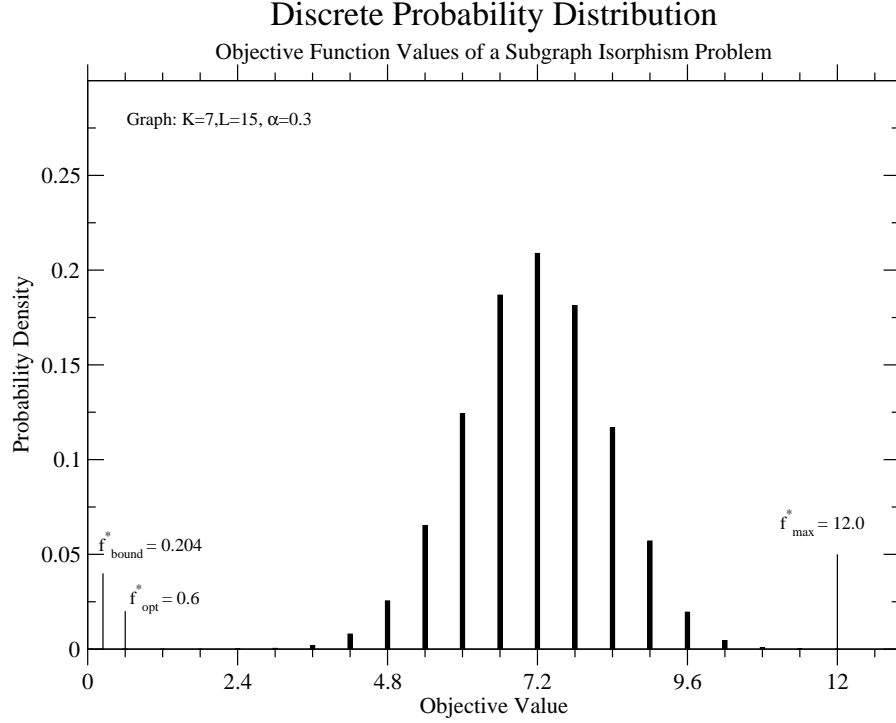


Figure 5.62: The distribution of the objective values for the subgraph isomorphism problem which is shown in figure 5.61. The objective values (5.7) for this kind of problem instances are restricted to discrete values, as the quadratic term $\alpha x^\top Q x$ can only attain values which are multiples of 2α . Here we have set α arbitrarily to 0.3. The optimal objective value is 0.6 and the obtained lower bound is $0.204 > 0.0$, which is a non-isomorphism proof for this problem instance.

no subgraph isomorphism can occur in this problem instances. The other 612 problem instances have a lower bound ≤ 0.0 . For 436 ($\approx 71\%$) of these problem instances the combinatorial optimum is > 0.0 indicating that the relaxation is not tight enough to detect that no subgraph isomorphism can occur.

But probably, the tightness and therefore the lower bound can be improved by reducing the dimension of the problem size (see also section 5.9.2). For example one can eliminate a mapping $i \mapsto j$ if the degree⁶ of an object node i is larger than the degree of node j in the scene graph. Such a mapping cannot lead to a subgraph isomorphism.

However, for increasing problem instances we have observed that the relaxation gets less tight and a lower bound ≤ 0.0 becomes more likely.

⁶The number of incident edges.

Chapter 6

Conclusion

6.1 Summary

In this thesis we investigated convex optimization techniques to approximate combinatorial problems which arise in computer vision in the context of view-based object recognition. In particular we were concerned with a weighted graph-matching approach and with a subgraph matching approach which may be utilized as a component of an object recognition system which internally represents the objects and scenes as graphs.

As the convex relaxation represents a central technique in this thesis to approximate these NP-hard problems we point out the advantages of this technique in the following:

- The original intractable integer optimization problem is approximated by an analytical convex mathematical optimization problem. Due to its convexity it can be optimally solved under mild conditions.
- No additional tuning parameters are involved in the optimization process which can critically influence the quality of the solution and must be optimized thoroughly.
- One can not be trapped in a local minimum in an early stage of the algorithm.
- There exists a performance bound for quadratic and semidefinite optimization problems which guarantees a combinatorial solution to be within the best $\frac{4}{7}$ of the total range of all objective values. But note that this performance bound is obtained by the random hyperplane method which we did not apply to obtain the combinatorial solutions.
- From the computational point of view the exponential complexity of calculating the combinatorial optimum is reduced to a polynomial complexity of calculating the convex relaxation which approximates the original problem.
- The solution of a convex relaxation provides a lower (upper) bound to the original combinatorial minimization (maximization) problem and thus can

be used as a subroutine within an exact search strategy like branch-and-bound. Note that such approaches have exponential complexity in the worst case. In our approaches we used the convex relaxation directly as approximation to the combinatorial solution.

A general method to obtain a convex relaxation is the Lagrange relaxation and for indefinite quadratic 0/1-optimization problems a recipe [115] can be used to obtain a (convex) semidefinite problem formulation. The connection between this recipe and the Lagrange relaxation is discussed in [95]. The convex relaxation of the weighted graph matching problem results in a convex quadratic program and the convex relaxation of our proposed subgraph matching approach results in a semidefinite program. An apparent drawback of the convex relaxation approaches for integer problems lies in the following fact:

- The solution of the convex relaxation is usually not a feasible solution for the original problem and a close combinatorial solution must be computed by a post-processing step. The obtained solution represents often only a suboptimal solution. However, experiments reveal that the obtained solutions are usually close to the global optimum and much better than a basic performance bound guarantees.

One has the following additional disadvantage in obtaining a semidefinite relaxation from a quadratic optimization problem:

- The number of variables is squared and often also the number of constraints increases very quickly. This reduces the size of problems one can cope with.

However, SDP relaxations have turned out to be extremely successful in approximating combinatorial problems [2, 68, 69, 134]. Also our investigations show that convex optimization methods and especially the semidefinite relaxation is an attractive direction of research and very good approximations for these NP-hard relational matching problems are obtained.

6.1.1 Weighted Graph Matching

In chapter 4 we studied a convex programming approach [21] to the quadratic assignment problem (QAP). In our context, the quadratic assignment problem [24] corresponds to the weighted graph matching of two graphs with an *equal* number of vertices. In this graph matching approach the matchings are represented by permutation matrices and we observed the following properties of the combinatorial approach:

- The solution of the combinatorial QAP graph matching approach results in the desired matching if the graphs are quite similar, which is of course a required characteristic for a useful graph matching approach.
- The objective function favors matchings which map large edges of the first graph to large edges in the second graph. Therefore, strong perturbations

which affect the large weights in the graphs are likely to lead to a combinatorial optimum which corresponds to an undesired matching. But from the viewpoint of computer vision large weights can be expected to involve reliable feature measurements. Therefore large weighted edges are likely to be present in both the object and scene graph.

- The QAP graph matching approach is invariant against rotation, translation and scaling of an image which makes it an attractive approach for object recognition tasks in computer vision.

The convex relaxation of this weighted graph matching approach was explained in detail and results in a convex quadratic program. According to the previously listed advantages of convex programming this approach does not involve any tuning parameter and the global solution can be computed in polynomial time by interior point methods [140]. To obtain a combinatorial solution from the convex relaxation we used the following post-processing step:

- A good local minimum was obtained by solving a linear program which computes a permutation matrix close to the approximation. A slight improvement of this approach was achieved by considering a linear approximation of the original problem which results in a linear program as well.

We compared the convex relaxation approach with a recent deterministic annealing approach [55, 77] and an approach based on the eigenvalue decomposition [139]. The comparison was based on several benchmark problems from the QAPLIB-collection [25] and on statistical results computed for a large set of randomly generated graphs which include ground truth experiments. Furthermore we compared some of our QAPLIB-collection results with results published in [120] which are obtained by a SDP relaxation of the quadratic assignment problem. We found the following:

- The performance of the approach based on the eigenvalue decomposition is worse than the convex quadratic relaxation approach whereas the deterministic annealing approach performs similarly or slightly better, but uses parameter values which were optimized thoroughly by hand.
- The results of the SDP relaxation indicates that the SDP approach performs in terms of the objective function better than the convex quadratic relaxation approach. But one has to keep in mind that the SDP relaxation squares the number of variables and therefore increases the computational effort to solve the relaxation.

Beside the statistical investigation, an example was presented which shows the applicability of the QAP graph matching approach to real world problems. Nevertheless, a basic assumption behind the QAP graph matching approach is that the two graphs have the same number of nodes. This is an unrealistic assumption for practical use as due to noise, occlusion or distortion the graphs

are likely to have a different number of vertices in computer vision applications. In order to cope also with subgraph matching problems we suggested two enhancements of the QAP graph matching approach:

- The first straight forward approach was to add dummy nodes to the smaller graph such that the resulting problem complies with the requirement of equally sized graphs. This approach turns out to result in unsatisfactory matchings.
- Secondly we proposed to comprise a convex correction term which represents an important adjustment within the combinatorial matching approach. The correction term originates from the original combinatorial matching approach when applied to differently sized graphs. This correction leads to the desired subgraph matchings.

Note that we have postponed the investigation of the latter approach. Nevertheless we outlined an appropriate convex formulation and showed how this problem can be solved using a gradient descent algorithm on a particular manifold and we think this approach is worth to be investigated in the future.

6.1.2 Subgraph Matching

In chapter 5 we proposed a new quadratic integer program to the problem of subgraph matching. We extended the linear programming formulation for computing optimal matchings in bipartite graphs by adding a quadratic term which comprises the relational constraints given by both graphs. The resulting quadratic integer minimization problem has the following properties:

- The objective function along with the constraints models the subgraph matching in an adequate way. That means that the optimal combinatorial solution is nearly always in accordance with the desired matching. This qualified our approach as a reasonable subgraph matching approach.
- The minimization of the objective function prefers matchings where similar nodes are mapped to each other and also the underlying structure of the object graph is preserved in the assigned subgraph of the scene graph.
- The approach makes only use of a similarity measure between the nodes of the two graphs and of the relational structures of both graphs which are given by their 0/1-adjacency matrices.
- The introduction of a regularization parameter α allows to adjust the influence of the structure related term compared to the similarity measure.
- The approach is invariant against rotation and translation. Depending on the similarity measure it is also scale invariant. These are properties which are favored for object recognition tasks that intend to detect objects regardless of the position, rotation and scaling of an object in an image.

One can identify situations where the approach is not applicable or must be modified to work:

- The approach is only reasonable for non-fully connected scene graphs as a fully connected scene graph contains no usable information about the structure. However, in such a case the problem formulation is equivalent to the linear bipartite matching problem which maps the locally best fitting nodes to each other.
- As all existing object graph nodes are mapped to the scene graph, occlusion of the object in the scene might become a problem. A possible way to handle occlusion could be to break the object graph down into several parts which are independently matched against the scene.

We explained in detail the (convex) SDP relaxation for this combinatorial subgraph matching approach. The previous discussed advantages of convex programs apply and the solution can be computed without the need of any additional parameter in polynomial time. To compute reasonable combinatorial solutions we proposed the following interpretation of the solution obtained by the SDP relaxation:

- Due to the weakly incorporated 0/1-integer constraints in the SDP relaxation the vector on the diagonal of the SDP solution matrix is interpreted as “non-integer” approximation for the combinatorial solution.
- According to the matching constraints which are incorporated into the SDP relaxation we interpret the element values in the non-integer solution vector as probability for the appropriate mapping. This assumption is strongly supported by our observation that indeed good matching candidates have higher values in the solution vector.

This probabilistic interpretation gives rise to the following consecutive post-processing approaches.

- First we proposed a linear optimization problem to compute a close combinatorial solution to the approximated solution which according to the probabilistic interpretation, represents the likeliest matching.
- The second post-processing step uses the likeliest matching as starting point and tries to improve this matching by exploiting the probabilistic information in the approximated solution. We proposed a sampling post-processing step which considers other possible matchings according to their probabilities as well. Note that this approach can not worsen the previous approach.

We performed numerous large experiments to investigate the performance of the SDP subgraph matching approach. The main results are summarized in the following:

- For smaller sized subgraph matching problems our SDP subgraph matching approach results nearly always in the combinatorial optimum. Even for larger sized problems an impressively large fraction of problems can be solved to optimality. Therefore, an important factor for the reliability

of our subgraph matching approach is the size of the problem data. This is not unexpected as the number of possible matchings increase exponentially with increasing graph size.

- Despite the exponential growth of the problem size with increasing graph size the tightness decreases only slightly and in terms of the objective value still very good solutions are obtained.
- The probabilistic interpretation of the solution is very reasonable as it is often able to improve the obtained solutions to optimality.

Furthermore our experiments reveal the dependence of the subgraph matching approach on “external” parameters:

- Problem creation parameters like the edge density and the quality of the similarity measure, thus external parameters, affects the tightness of the SDP approximation and therefore the performance of the SDP subgraph matching approach to result in the global optimum.

Several real world examples show that our SDP subgraph matching approach is a very reasonable approach for graph matching in computer vision applications. It seems that such problems are good natured and usually the desired matching shows up very clearly in the approximation. As real world subgraph matching problems tend to be large we proposed two possible ways to reduce the computational effort:

- The first approach reduces the number of constraints by combining several gangster constraints into a single constraint but has the disadvantage that the relaxation becomes less tight. A grouping of up to four constraints seems to be reasonable as it results in a strong computational gain without losing too much accuracy.
- The second approach reduces the problem size by eliminating undesired mappings directly from the problem instance. However, this approach includes the danger that desired mappings are accidentally eliminated from the original problem.

Maybe a combination of these two techniques can be used to further improve the combinatorial solution. In particular a constraint reduced solution could be used to determine the unlikeliest mappings that can be eliminated as undesired mappings from the original problem. Some iterations of this process will scale down the problem size such that it can be solved to optimality. Hopefully this smaller problem still includes the optimal matching of the original problem.

Some advanced investigations showed the following interesting facts:

- The SDP subgraph matching approach can cope with small structural perturbations and is insensitive against errors in the similarity measure. However, an increased perturbation level raised the likelihood that an undesired matching is obtained.

- A bimodal experiment shows that the solution vector comprises the knowledge of two potentially good mappings in the form of higher probabilities for both matchings. Thus, we answer the question of how bi- or multimodal optimization problems are reflected in the convex relaxation which is by definition only unimodal.
- Another very interesting observation is that our SDP relaxation can serve as bound to the subgraph non-isomorphism problem between two simple graphs. Nevertheless, the latter finding is in a preliminary stage and worth to be investigated more thoroughly.

To summarize, we have proposed a very promising combinatorial subgraph matching approach along with a convex approximation. The presented statistical and real world results show that this subgraph matching approach is a considerable choice for subgraph matching problems that occur in computer vision.

6.2 Future Work

We have seen that convex relaxation is a very promising approximation technique for the graph and subgraph matching approaches discussed in this thesis, especially if one considers the combinatorial nature of these integer programs. Our results showed that this technique along with an appropriate post-processing is very likely to result in good combinatorial solutions. Indeed today the main topic for semidefinite programming is the approximation of combinatorial problems like the quadratic knapsack problem [68], the max-cut problem [95], partitioning, grouping, and image restoration problems [83] to name a few. This variety of applications encourages to find further computer vision problems that could be approximated tightly by convex relaxations. Currently an ongoing area of research is the development of efficient algorithms for solving SDP programs [2, 146, 69, 13].

In the following we discuss some graph matching related topics for future work. We start with particular ideas we proposed in this thesis but which were not yet implemented or investigated. Then we propose wider topics in the context of graph matching for future investigations.

QAP Graph Matching

With regard to the QAP graph matching approach there are the following aspects that could be investigated:

- For the SDP subgraph matching approach discussed in chapter 5 we proposed a probabilistic interpretation of the solution computed by the SDP relaxation. Nearly the same interpretation can be applied to the approximated solutions obtained by the convex quadratic relaxation of the QAP graph matching approach. This interpretation gives rise to propose an adapted two-opt post-processing which exchanges in each iteration two

mapping pairs according to their probability instead of uniformly selected pairs. We think that this minor modification could be a fruitful supplement for this post-processing step and likely results in improved objective values.

- Furthermore, we have not yet investigated the promising enhancement of the QAP graph matching approach which is able to cope also with weighted subgraph matching problems. Note that we already proposed a convex approximation for this approach. The implementation and investigation will show whether this approach can indeed deal with (weighted) subgraph matching problems.

SDP Subgraph Matching

Very interesting future work is related to our SDP subgraph matching approach:

- We proposed in section 5.9.5 a bound for non-subgraph-isomorphism. A positive bound for a subgraph matching instance represents the proof that no subgraph isomorphism can be found in the particular problem instance. Moreover we are interested whether the solution of the convex relaxation can be utilized to detect a present subgraph isomorphism. So far, this topic is only preliminary investigated but we believe it is worth to investigate the capability of this bound thoroughly.

Object Representation

The investigation of our graph and subgraph matching approaches in the context of computer vision lead direct to the question how objects and therefore image information can be represented as graphs in a reliable way ?

- One can find some papers concerned with this subject e.g in [79] or [61] which are publications related to a IAPR workshop “Graph Based Representations in Pattern Recognition”. The superiority of a graph based representation is pointed out in [37]. But due to computational reasons often the relational representation in object recognition tasks is confined to tree structured representations like the shock trees in [131] or the model graphs used in [41].

However, the development of good approximation algorithms for graph matching and the increasing computational power qualifies graph based approaches for the application in computer vision tasks. This gives raise to the assumption that the use of relational representations will increase over the time.

The question of which features have to be chosen as part of the relational representation of an object has been deliberately excluded from this work, because the range of the literature and corresponding results is extremely broad. Nonetheless, some general conclusions can be drawn in this connection, based on the findings reported in this thesis.

Firstly, it has been shown that the matching approach is more sensitive to structural perturbations of relational object views than to noise in the similarity measurements. Secondly, the matching problem can be successfully solved for structured graphs - as opposed to almost complete graphs - because the relaxation then is very tight. Finally, the approach can cope with ambiguities in terms of *several* good matchings.

As a result, the selection of features for representing object views should primarily focus on structural stability of the feature extraction stage under realistic imaging conditions of the application. Furthermore, it makes sense to restrict object representations to *sparse* graphs, at the cost ambiguities in complex scenes. The latter can be dealt with by considering not only the best match, but other probable matchings provided in *the same* computational step, too.

The development of an feature extraction scheme directly combined with the matching approach investigated in this thesis and a corresponding recognition criterion is a worthwhile future research topic.

Object Recognition

The question how the graph matching approaches can be utilized in object recognition tasks was beyond the scope of this work but gives rise to investigate this subject in the future:

- The empirical evaluation in section 5.6 and 5.7 clearly showed that values of the objective function for successful matchings markedly deviate from expected values of arbitrary matchings. For specific scenes, this property may already turn out to be sufficient for recognizing objects. More sophisticated schemes for verifying the presence of an object raise interesting questions for further research. Irrespective of how such a scheme may look like, the set of most probable matchings, computed with the approach investigated in this thesis, will provide a suitable starting point.

Performance Guarantees

The performance results of our subgraph matching in this thesis gives rise to the question if one can prove a performance guarantee for our SDP approximation algorithm.

- The idea to use randomized rounding to study approximation algorithms was published first by Raghavan and Thompson [117]. Goemans and Williamson [54] proposed a randomized approximation algorithm with a good performance guarantee for maximum-cut problems which was based on a semidefinite relaxation followed by the *randomized hyperplane technique* to obtain a feasible combinatorial solution. In particular they were able to proof for the maximum-cut problem (a maximization problem) that the expected solution value of their approach is at least 0.87856 times the optimal value. Since then the *randomized hyperplane technique* has

been applied to several semidefinite relaxations of combinatorial problems in order to obtain feasible integer solutions. These include the maximum directed cut problem [43], the maximum k -cut problem [49], machine scheduling problems [133], the quadratic knapsack problem [68] and others (see e.g. [146]). Eventually similar randomized rounding techniques can be applied to obtain a performance guarantee for our SDP subgraph matching approach where the feasible combinatorial solution must represent a matching.

Appendix A

Quadratic Assignment Supplements

A.1 The Dual of the relaxed homogeneous QAP

Consider the following orthonormal constrained optimization problem, which represents a relaxation of the homogeneous quadratic assignment problem. $A \in \mathbb{R}^{x \times n}$ and $B \in \mathbb{R}^{x \times n}$ are symmetric matrices and $X \in \mathbb{R}^{x \times n}$ is an orthonormal matrix ($X \in O$):

$$\begin{aligned} \min_X \quad & \text{Tr}[AXB^\top X^\top] \\ \text{s.t.} \quad & XX^\top = I \\ & X^\top X = I \end{aligned} \tag{A.1}$$

We outline how the Lagrangian dual of the primal optimization problem (A.1) can be calculated. Further we explain in section A.1.1 how the dual of (A.1) can be stated as a linear optimization problem. The constraints of (A.1) can be written element wise as:

$$\begin{aligned} -(XX^\top)_{ab} + I_{ab} &= 0 \quad a, b = 1, \dots, n \\ -(X^\top X)_{cd} + I_{cd} &= 0 \quad c, d = 1, \dots, n \end{aligned}$$

For the two different constraint sets the Lagrange multiplier S_{ab} , $a, b = 1, \dots, n$ and T_{cd} , $c, d = 1, \dots, n$ are introduced to obtain the Lagrangian of (A.1). Considering now only the sum related to the Lagrange multipliers S_{ab} in the Lagrangian we find the following trace formulation:

$$\begin{aligned} -\sum_{a,b}^{n,n} (S^\top)_{ba} ((XX^\top)_{ab} - I_{ab}) &= \\ -\sum_a^n ([S^\top(XX^\top)]_{aa} - [S^\top I]_{aa}) &= -\text{Tr}[S^\top(XX^\top - I)] \end{aligned}$$

Here S is symmetric ($S = S^\top$) as can be seen from

$$-\text{Tr}[S^\top(XX^\top - I)] = -\text{Tr}[(XX^\top - I^\top)S^\top] = -\text{Tr}[S(XX^\top - I)]$$

Together with a similar calculation for the second constraints ($-X^\top X + I = 0$) the Lagrangian is:

$$L(X, S, T) = \text{Tr}[AXB^\top X^\top] - \text{Tr}[S(XX^\top - I)] - \text{Tr}[T(X^\top X - I)] \quad (\text{A.2})$$

With the identity: $\text{Tr}[KXL^\top X^\top] = \text{vec}(X)^\top (L \otimes K) \text{vec}(X)$ (cf. (4.20)) the dual function $w(S, T) = \min_X L(X, S, T)$ is then:

$$\begin{aligned} w(S, T) &= \min_X \text{Tr}[AXB^\top X^\top] - \text{Tr}[S(XX^\top - I)] - \text{Tr}[T(X^\top X - I)] \\ &= \min_X \text{Tr}[AXB^\top X^\top] - \text{Tr}[SXI X^\top] + \text{Tr}[S] - \text{Tr}[TX^\top IX] + \text{Tr}[T] \\ &= \min_X \text{vec}(X)^\top (B \otimes A) \text{vec}(X) - \text{vec}(X)^\top (I \otimes S) \text{vec}(X) + \text{Tr}[S] + \\ &\quad - \text{vec}(X)^\top (T \otimes I) \text{vec}(X) + \text{Tr}[T] \\ &= \min_X \text{vec}(X)^\top [(B \otimes A) - (I \otimes S) - (T \otimes I)] \text{vec}(X) + \text{Tr}[S] + \text{Tr}[T] \end{aligned}$$

The dual problem $\max_{S, T} w(S, T)$ reads now:

$$\max_{S, T} w(S, T) = \max_{S, T} \min_X \text{vec}(X)^\top [(B \otimes A) - (I \otimes S) - (T \otimes I)] \text{vec}(X) + \text{Tr}[S] + \text{Tr}[T] \quad (\text{A.3})$$

The inner minimization of the dual problem (A.3) is only bounded, if $(B \otimes A) - (I \otimes S) - (T \otimes I) \succeq 0$ is positive semidefinite. Making these constraints explicit we arrive at the following dual problem:

Dual:

$$\begin{aligned} \max_{S, T} \quad & \text{Tr}[S] + \text{Tr}[T] \\ \text{s.t.} \quad & [(B \otimes A) - (I \otimes S) - (T \otimes I)] \succeq 0 \end{aligned} \quad (\text{A.4})$$

The dual problem (A.4) can be stated as the following linear optimization problem

$$\begin{aligned} \max \quad & e^\top s + e^\top t \\ \text{s.t.} \quad & t_i + s_j \leq c_{ij} = \lambda_i \sigma_j \end{aligned} \quad (\text{A.5})$$

which is shown in the next section.

A.1.1 QAP Dual as linear program

The aim of this section is to show the equivalence of the dual problem (A.4) and the linear optimization problem (A.5). From the stationary condition of the Lagrangian (A.2) at the optimum, one finds

$$\frac{\partial}{\partial X} L(X, S, T) = AXB^\top - SX - XT = 0.$$

which is equivalent to:

$$S + XTX^\top = AXB^\top X^\top \quad (\text{A.6})$$

If we transpose the arguments of the traces in the Lagrange function (A.2), which does not change the objective function, we have

$$\bar{L}(X, S, T) = \text{Tr}[XB^\top X^\top A] - \text{Tr}[S^\top (XX^\top - I)] - \text{Tr}[T^\top (X^\top X - I)] \quad (\text{A.7})$$

By setting the derivation of the Lagrange function to zero, we obtain

$$\frac{\partial}{\partial X} \bar{L}(X, S, T) = AXB^\top - S^\top X - XT^\top = 0,$$

which is equivalent to:

$$S^\top + XT^\top X^\top = AXB^\top X^\top \quad (\text{A.8})$$

Observing that the right sides of the equations (A.6) and (A.8) are equal, we get:

$$S + XTX^\top = S^\top + XT^\top X^\top = (S + XTX^\top)^\top \quad (\text{A.9})$$

Further by inserting (A.6) into (A.9) one finds

$$AXB^\top X^\top = XB^\top X^\top A,$$

from which we see that A and $XB^\top X^\top$ commute. From this we conclude that these two terms can be diagonalized simultaneously by an orthogonal matrix V . From (A.6) we get:

$$\underbrace{V^\top AV}_{\Sigma} \underbrace{V^\top XB^\top X^\top V}_{\bar{\Lambda}} = \Sigma \bar{\Lambda} = V^\top (S + XTX^\top) V$$

We assume that the matrix A is diagonalized by $V^\top AV = \Sigma$ and that B is diagonalized by the orthogonal transformation $U^\top BU = \Lambda$. The matrices Λ , $\bar{\Lambda}$ and Σ are diagonal matrices. Substituting $B = U\Lambda U^\top$ in the diagonalization of $XB^\top X^\top$:

$$V^\top XB^\top X^\top V = \bar{\Lambda},$$

we get

$$\underbrace{\underbrace{V^\top XU}_P \underbrace{\Lambda U^\top X^\top V}_{P^\top}}_{\bar{\Lambda}} = P\Lambda P^\top = \bar{\Lambda}$$

The two diagonal matrices Λ and $\bar{\Lambda}$ can only be different by the order of eigenvalues on the diagonal. We conclude that $V^\top XU$ must be a permutation matrix $P \in \Pi$:

$$V^\top XU = P$$

With $V^\top X = PU^\top$ we get

$$V^\top(S + XTX^\top)V = V^\top SV + V^\top XTX^\top V = V^\top SV + PU^\top TUP^\top = \Sigma \bar{\Lambda}$$

which represents a diagonal matrix. We see that $V^\top SV = \bar{S}$ and $U^\top TU = \bar{T}$ must be diagonal matrices. The permutation matrix P in the term related to T just reorders the diagonal entries in the diagonal matrix $U^\top TU$.

To summarize we have the following valid diagonalizations:

$$\begin{aligned} U^\top BU &= \Lambda \quad , \quad U^\top TU = \bar{T} \\ V^\top AV &= \Sigma \quad , \quad V^\top SV = \bar{S} \end{aligned}$$

With that the dual problem (see also (A.4))

$$\begin{aligned} \max_{S, T} \quad & \text{Tr}[S] + \text{Tr}[T] \\ \text{s.t.} \quad & [(B \otimes A) - (I \otimes S) - (T \otimes I)] \succeq 0 \end{aligned}$$

can be written as linear problem. First the positive semidefinite constraint can be rewritten as:

$$U\Lambda U^\top \otimes V\Sigma V^\top - UIU^\top \otimes V\bar{S}V^\top - U\bar{T}U^\top \otimes VIV^\top \succeq 0$$

We can factor $(U \otimes V)$ and $(U^\top \otimes V^\top)$ out and obtain:

$$(U \otimes V)(\Lambda \otimes \Sigma - I \otimes \bar{S} - \bar{T} \otimes I)(U^\top \otimes V^\top) \succeq 0$$

This is equivalent to

$$(\Lambda \otimes \Sigma - I \otimes \bar{S} - \bar{T} \otimes I) \succeq 0$$

where the matrices have only diagonal elements. With $\text{Tr}[S] = \text{Tr}[V\bar{S}V^\top] = \text{Tr}[\bar{S}V^\top V] = \text{Tr}[\bar{S}]$ and similar $\text{Tr}[T] = \text{Tr}[\bar{T}]$ the dual problem can be written as:

$$\begin{aligned} \max_{\bar{S}, \bar{T}} \quad & \text{Tr}[\bar{S}] + \text{Tr}[\bar{T}] \\ \text{s.t.} \quad & [\Lambda \otimes \Sigma - I \otimes \bar{S} - \bar{T} \otimes I] \succeq 0 \end{aligned} \tag{A.10}$$

Recognizing that the matrices have only diagonal entries we get the desired result that this problem is equivalent to the following linear optimization problem:

$$\begin{aligned} \max_{s, t} \quad & e^\top \bar{s} + e^\top \bar{t} \\ \text{s.t.} \quad & (\lambda_i \sigma_j - \bar{s}_j - \bar{t}_i) \geq 0 \quad i, j = 1, \dots, n \end{aligned} \tag{A.11}$$

A.1.2 Non unique solutions for the EVB

The Eigenvalue Bound

$$(EVB) \quad \min_{X \in \mathcal{O}} \text{Tr}[AXB^\top X^\top] = \langle \lambda(A), \lambda(B) \rangle_-$$

is obtained at least for $X = UW^\top$ where U and W diagonalize the symmetric matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ respectively and further sorts the eigenvalues of A and B^\top in an appropriate order (see section 4.2.2).

We show that the minimum is also obtained for $\tilde{X} = UDW^\top$ where D is a diagonal matrix with diagonal elements $D_i = \{1, -1\}$.

As the trace is invariant against a cyclic rotation of the matrices in the argument we are searching for additional solutions \tilde{X} beside X where the following holds true:

$$\begin{aligned} AXB^\top X^\top &= A\tilde{X}B^\top \tilde{X}^\top \\ X^\top AXB^\top &= \tilde{X}^\top A\tilde{X}B^\top \end{aligned}$$

Assuming A and B have full rank this is equivalent with

$$\begin{aligned} XB^\top X^\top &= \tilde{X}B^\top \tilde{X}^\top \\ X^\top AX &= \tilde{X}^\top A\tilde{X}. \end{aligned} \tag{A.12}$$

As X and \tilde{X} are orthogonal matrices we can introduce Y and Z as follows:

$$\begin{aligned} Z &= X^{-1}\tilde{X} = X^\top \tilde{X} \\ Y &= \tilde{X}X^{-1} = \tilde{X}X^\top. \end{aligned}$$

Using this we get from (A.12):

$$\begin{aligned} B^\top &= ZB^\top Z^\top \\ A &= Y^\top AY. \end{aligned}$$

We can see that $B^\top Z = ZB^\top$ and $AY = YA$ which means that the pairs B^\top, Z and the A, Y are simultaneously diagonalizable. B^\top is diagonalized by W therefore Z is also diagonalized by W . The same is true for A and Y , which can be diagonalized by U .

$$\begin{aligned} Z &= X^\top \tilde{X} = WD_ZW^\top \\ Y &= \tilde{X}X^\top = UD_YU^\top \end{aligned} \tag{A.13}$$

Here D_Z and D_Y are diagonal matrices. Inserting $X = UW^\top$ in (A.13) we find,

$$\begin{aligned} \tilde{X} &= UD_ZW^\top \\ \tilde{X} &= UD_YW^\top \end{aligned}$$

from which we conclude that $D_Z = D_Y = D$. From

$$\begin{aligned} I &= \tilde{X}\tilde{X}^\top = UDW^\top WD^\top U^\top \\ &= UDD^\top U^\top = UDDU^\top \end{aligned}$$

we find that $DD = I$. Therefore we have $D_i^2 = 1$ for the diagonal elements and so $D_i = \{+1, -1\}$. That means that one can obtain 2^n solutions \tilde{X} by using $\tilde{X} = UDW^\top$ where D is a diagonal matrix with $\{1, -1\}$ values on the diagonal.

A.2 QAP-Bounds

We list several more results computed for problems in the QAPLIB-collection [25]. In table A.1 the optimum f^* , the eigenvalue bound EVB (4.10), the projected eigenvalue bound $PEVB$ (4.14) and the quadratic programming bound QPB (4.23) are shown.

Problem	f^*	EVB	$PEVB$	QPB	SDP
chr12a	9552	-135327	-21886	-17789	n.a.
chr12b	9742	-136734	-17099	-13526	n.a.
chr15b	7990	-196657	-54711	-49179	n.a.
chr18a	11098	-241984	-68211	-60980	n.a.
chr18b	1534	-10945	-902	-673	n.a.
chr20a	2192	-32762	-8392	-7590	n.a.
chr20c	14142	-340384	-87222	-75497	n.a.
chr22a	6156	-67199	-21313	-19750	n.a.
chr25a	3796	-70912	-22351	-20950	n.a.
esc16a	68	-149	47	47	59
esc16d	16	-124	-19	-19	8
esc16e	28	-123	6	6	23
esc16g	26	-138	9	9	20
esc16i	14	-153	-25	-25	9
esc16j	8	-77	-6	-6	7
esc32a	130	-926	-150	-150	n.a.
esc32b	168	-704	65	65	n.a.
esc32c	642	-1021	464	464	n.a.
esc32d	200	-502	98	98	n.a.
esc32e	2	-469	-165	-165	n.a.
esc32f	2	-469	-165	-165	n.a.
esc32g	6	-286	-76	-76	n.a.
esc32h	438	-864	245	245	n.a.
esc64a	116	-993	-243	-243	n.a.
had12	1652	-1407	1573	1586	1643
had14	2724	-2488	2609	2628	2715
had18	5358	-4422	5104	5137	5317
had20	6922	-5785	6625	6673	6885
kra30a	88900	-154081	63717	67151	77647
nug12	578	-909	472	481	557
nug14	1014	-1505	871	884	992
nug15	1150	-1745	973	990	1122
nug16a	1610	-2184	1403	1436	1570
nug16b	1240	-1806	1046	1061	1188
nug17	1732	-2397	1487	1518	1669
nug18	1930	-2566	1663	1694	1852
nug20	2570	-3198	2196	2232	2451
nug21	2438	-3992	1979	2038	2323
nug22	3596	-6109	2966	3076	3440
nug24	3488	-4917	2960	3015	3310
nug25	3744	-4725	3190	3269	3535
nug30	6124	-7836	5266	5342	5803
scr12	31410	-134544	4727	9023	29321
scr15	51140	-206336	10355	16259	47840
scr20	110030	-454501	16113	28118	94998
sko64	48498	-43170	43890	44492	n.a.
sko72	66256	-57703	60402	61088	n.a.
ste36a	9526	-87952	-11771	-10523	n.a.
ste36b	15852	-427073	-130008	-125755	n.a.
ste36c	8239110	-71578370	-8965412	-7462533	n.a.

Table A.1: Bounds computed for QAPLIB-problems with rank deficit.

The available bounds obtained by a semidefinite relaxation approach (cf. section 4.3.3) are taken from [120].

Appendix B

Subgraph Matching Supplements

B.1 Example Data

To enable the reader to reproduce our results we list for two smaller subgraph matching problems discussed in chapter 5 the appropriate data. Note that only the adjacency matrices N_K , N_L and the similarity vector w are used within our SDP subgraph matching approach but for the sake of completeness also the coordinates of the model nodes and scene nodes are given.

B.1.1 Illustrative Example Data

We first list the data for the small illustrative subgraph matching experiment which was discussed in section 5.5. The problem is for example depicted in figure 5.1).

The adjacency matrices of the two graphs with sizes $K = 5$ and $L = 13$ are:

$$N_K = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} N_L = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

The similarity vector w is printed below where the elements are arranged as

specified in section 2.2.2:

$$w = (0.882, 0.831, 0.880, 0.904, 0.808, 0.996, 0.814, 0.966, 0.524, 0.839, 0.844, 0.909, 0.966, \\ 0.200, 0.880, 0.971, 0.802, 0.935, 0.935, 0.948, 0.915, 0.810, 0.481, 0.880, 0.833, 0.979, \\ 0.929, 0.975, 0.824, 0.982, 0.915, 0.809, 0.863, 0.200, 0.871, 0.899, 0.505, 0.959, 0.990, \\ 0.928, 0.895, 0.824, 0.200, 0.979, 0.979, 0.813, 0.844, 0.899, 0.946, 0.834, 0.485, 0.923, \\ 0.921, 0.200, 0.998, 0.914, 0.858, 0.907, 0.927, 0.814, 0.960, 0.948, 0.936, 0.885, 0.510)^\top$$

The following matrices V_K and V_L contain the positions of the nodes of the two graphs G_K and G_L :

$$V_K = \begin{pmatrix} 0.50 & 0.50 \\ 0.50 & 0.60 \\ 0.55 & 0.70 \\ 0.60 & 0.60 \\ 0.60 & 0.50 \end{pmatrix} V_L = \begin{pmatrix} 0.42 & 0.76 \\ 0.64 & 0.41 \\ 0.71 & 0.69 \\ 0.43 & 0.50 \\ 0.73 & 0.60 \\ 0.30 & 0.65 \\ 0.40 & 0.64 \\ 0.32 & 0.77 \\ 0.50 & 0.50 \\ 0.50 & 0.60 \\ 0.55 & 0.70 \\ 0.60 & 0.60 \\ 0.60 & 0.50 \end{pmatrix}$$

In these matrices the first column represent the x-coordinates and the second the y-coordinates of the graph nodes. Furthermore, the row number specifies the label of the node which has the coordinates given by that row.

B.1.2 Graph Data for the Parameter Dependency Example

The second data we list belongs to subgraph matching experiment in section 5.6.3 which was used to visualize the parameter dependency. The subgraph matching problem is shown figure 5.16 and the adjacency matrices of these graphs are:

$$N_K = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} N_L = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The similarity vector w is:

$$w = (0.898, 0.903, 0.766, 0.857, 0.799, 0.958, 0.784, 0.711, 0.776, 0.776, 0.709, 0.914, \\ 0.753, 0.836, 0.822, 0.992, 0.771, 0.763, 0.794, 0.784, 0.672, 0.775, 0.641, 0.813, \\ 0.985, 0.976, 0.946, 0.685, 0.960, 0.889, 0.975, 0.703, 0.722, 0.751, 0.638, 0.893, \\ 0.848, 0.801, 0.646, 0.526, 0.785, 0.673, 0.977, 0.993, 0.642, 0.725, 0.733, 0.822, \\ 0.895, 0.791, 0.983, 0.922, 0.907, 0.875, 0.949, 0.895, 0.947, 0.887, 0.760, 0.898, \\ 0.976, 0.869, 0.626, 1.000, 0.738, 0.655, 0.770, 0.780, 0.856, 0.941, 0.840, 0.809)^\top$$

The coordinate matrices V_K and V_L for the nodes of the two graphs G_K and G_L are:

$$V_K = \begin{pmatrix} 0.163 & 0.690 \\ 0.157 & 0.116 \\ 0.143 & 0.100 \\ 0.173 & 0.162 \\ 0.158 & 0.187 \\ 0.125 & 0.147 \end{pmatrix} \quad V_L = \begin{pmatrix} 0.163 & 0.690 \\ 0.157 & 0.116 \\ 0.143 & 0.100 \\ 0.173 & 0.162 \\ 0.158 & 0.187 \\ 0.125 & 0.147 \\ 0.218 & 0.132 \\ 0.880 & 0.199 \\ 0.244 & 0.166 \\ 0.880 & 0.131 \\ 0.210 & 0.105 \\ 0.247 & 0.730 \end{pmatrix}$$

B.2 Earth Movers Distance

A distance between two distributions can be calculated by the so called Earth Mover's Distance (EMD) which was introduced by Rubner et. al. [123]. An intuitive way to interpret this distance measure is to define one distribution as earth-hills distributed in space and the other as holes in the same space. The holes of the second distribution should be filled with the earth of the first distribution. The used amount of work to fill a hole is defined by the product of the moved earth and the distance between the hill and the hole. The Earth Mover's Distance is then the least amount of work that is needed to fill the holes with the earth.

In the following we assume to have the two histograms $H = \{p_i, w_{pi}\}, i = 1, \dots, n$ and $K = \{q_j, w_{qj}\}, j = 1, \dots, n$ where p_i and q_j define the positions of the histogram columns and w_{pi} and w_{qj} are the appropriate column weights. An example for two such histograms with $n = 4$ and $m = 5$ is depicted in figure B.1.

To compute the Earth Mover's Distance a distance vector $d = (d_{11}, \dots, d_{mn})^\top$ is defined where d_{ij} is the distance between the two histogram columns p_i and q_j of the histograms.

$$d_{ij} = |p_i - q_j|$$

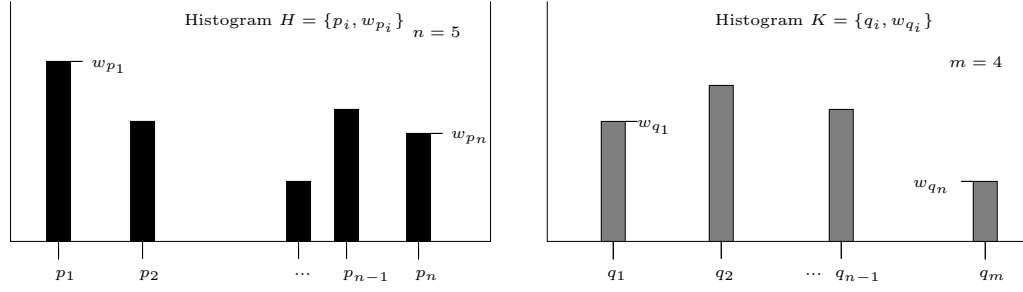


Figure B.1: An example for two histograms with $n = 4$ and $m = 5$ which should be compared by the Earth Mover's Distance.

Furthermore the flow vector $f = (f_{11}, \dots, f_{mn})^\top$ is introduced, where f_{ij} is the amount of material send from the column i of histogram H to the column j of the histogram K . The aim to find the flow vector f^* which minimizes the following costs:

$$\min_f \sum_{i=1}^m \sum_{j=1}^n d_{ij} f_{ij} = \min_f d^\top f$$

Note that the total cost is the sum of the moved material multiplied by the distance. The minimization is subject to $f_{ij} \geq 0$ which means that the material is only moved from histogram H to K and not vice versa.

The following three constraints limits the amount of material that can flow.

$$\sum_{j=1}^n f_{ij} \leq w_{p_i} \quad (\text{B.1})$$

$$\sum_{i=1}^m f_{ij} \leq w_{q_j} \quad (\text{B.2})$$

and

$$\sum_{i=1}^m \sum_{j=1}^n f_{ij} = \min\left(\sum_{i=1}^m w_{p_i}, \sum_{j=1}^n w_{q_j}\right) \quad (\text{B.3})$$

For example no more than the available material of a column can be transfered to a hole (B.1) and a hole can only be filled up until it is full (B.2). Therefore the total flow must be restricted to the available material or to the available space in the holes depending on which is lower (B.3).

Such a linear optimization problem can be efficiently calculated by any linear programming solver. The optimal flow for the example distributions is shown in figure B.2. The Earth Mover's Distance is defined as the total cost normalized by the total flow:

$$EMD = \frac{\sum_{i=1}^m \sum_{j=1}^n d_{ij} f_{ij}}{\sum_{i=1}^m \sum_{j=1}^n f_{ij}}$$

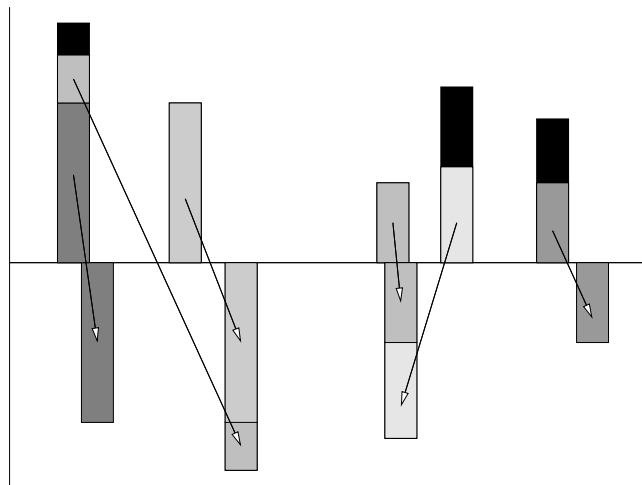


Figure B.2: Visualisation of the Earth Mover's Distance between two histograms. The histogram with $m = 4$ is interpreted as a distribution of holes while the histogram with $n = 5$ is interpreted as distribution of earth hills. The cheapest flow to fill the holes with earth is depicted by arrows.

We use this distance to compute the similarity between grey-value-histograms of image parts.

Bibliography

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
- [2] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.
- [3] Eric Allender, Michael C. Loui, and Kenneth W. Regan. Reducibility and completeness. Technical report, This material was written for Chapter 28 of the CRC Handbook of Algorithms and Theory of Computation, edited by Mikhail Atallah, 1998.
- [4] A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Poppiestone. A versatile computer-controlled assembly system. *Artificial Intelligence*, 6(2):129–156, 1975.
- [5] K. Anstreicher and H. Wolkowicz. On langrangian relaxation of quadratic matrix constraints. *SIAM J. Matrix Anal. Appl.* to appear.
- [6] K. M. Anstreicher and N. W. Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3):341–357, 2001.
- [7] K.M. Anstreicher and H. Wolkowicz. On lagrangian relaxation of quadratic matrix constraints. Technical report, Department of Combinatorics and Optimization, University of Waterloo, Ontario, Canada, 1998.
- [8] Mikhail J. Atallah. *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999.
- [9] H. G. Barrow and R.J. Popplestone. Relational descriptions in picture processing. *Machine Intelligence*, 6:377–396, 1971.
- [10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming – Theory and Algorithms*. John Wiley and Sons, New York. 25, second edition edition, 1993.
- [11] R. Bellman and K. Fan. On systems of linear inequalities in hermitian matrix variables. *In Proceedings of Symposia in Pure Mathematics, AMS*, 7, 1963.

- [12] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization, Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization. SIAM/MPS, 2001.
- [13] S.J. Benson and Y. Ye. DSDP3: dual scaling algorithm for general positive semidefinite programming. Preprint ANL/MCS-P851-1000, Argonne National Labs, 2001.
- [14] M. Berkelaar, J. Dirks, K. Eikland, P. Notebaert, and H. Schwab. Lp-solve software site. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.
- [15] Dimitri P. Bertsekas, Angelia Nedic, and Asuman E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.
- [16] S. Boettcher and A. G. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275–286, 2000.
- [17] I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.
- [18] B. Borchers. CSDP: A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.
- [19] Brian Borchers. *CSDP 4.8 User's Guide*, 2004.
- [20] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [21] N. Brixius and K. Anstreicher. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16(1–4):49–68, 2001.
- [22] Nathan William Brixius. *Solving Large-Scale Quadratic Assignment Problems*. PhD thesis, Graduate College of the university of Iowa, 2000.
- [23] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9):917 – 922, 1999.
- [24] R.E. Burkard, E. Çela, P.M. Pardalos, and L.S Pitsoulis. The quadratic assignment problem. In P.M. Pardalos and D.-Z. Du, editors, *Handbook of Combinatorial Optimization*, pages 241–338. Kluwer Acad. Publishers, 1998.
- [25] R.E. Burkard, S. Karisch, and F. Rendl. Qaplib – a quadratic assignment problem library. *J. Global Optimization*, 10:391–403, 1997.
- [26] Robert Carter and Kihong Park. How good are genetic algorithms at finding large cliques: an experimental study. Technical Report BU-CS-93-015, Computer Science Dept. Boston University, 1993.

- [27] L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989.
- [28] W. Christmas, Kittler. J, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 17(8):749–764, 1995.
- [29] Alan Cobham. The intrinsic computational difficulty of functions. *Proceedings of the 1964 Congress for Logic, Mathematics, and Philosophy of Science, North-Holland, Amsterdam*, pages 24–30, 1964.
- [30] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004.
- [31] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [32] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley, New York, 1998.
- [33] A.D.J. Cross and E.R. Hancock. Graph matching with a dual-step em algorithm. *IEEE Trans. Patt. Anal. Mach. Intell.*, 20(11):1236–1253, 1998.
- [34] A.D.J. Cross, R.C. Wilson, and E.R. Hancock. Inexact graph matching using genetic search. *Pattern Recog.*, 30(6):953–970, 1997.
- [35] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.
- [36] A. Dempster, N. Laird, and R. Rubin. Maximum likelihood estimation from incomplete data via the em algorithm. *Journal of Royal Statistical Society, Series B*, 39:1–38, 1977.
- [37] S. Dickinson. Object representation and recognition. In E. Lepore and Z. Pylyshyn, editors, *What is Cognitive Science?* Basil Blackwell publishers, 1999.
- [38] A. Edelman, T.A Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix. Anal.*, 20(2):303–353, 1998.
- [39] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.
- [40] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.
- [41] Dirk Farin, Peter H. N. de With, and Wolfgang Effelsberg. A segmentation system with model assisted completion of video objects. In Touradj Ebrahimi and Thomas Sikora, editors, *Proc. Conf. Visual Communications and Image Processing, Proc. SPIE 5150*, pages 366–377, Jun 2003.

- [42] O.D. Faugeras and M. Berthod. Improving consistency and reducing ambiguity in stochastic labeling: an optimization approach. *IEEE Trans. Patt. Anal. Mach. Intell.*, 3(4):412–424, April 1981.
- [43] U. Feige and M. X. Goemans. Approximating the value of two-prover proof systems, with applications to max 2sat and max dicut. *Proc. 3rd Israel Symposium on the Theory of Computing and Systems*, pages 182–189, 1995.
- [44] G. Finke, R.E. Burkard, and F. Rendl. Quadratic assignment problems. *Annals of Discrete Mathematics*, 31:61–82, 1987.
- [45] M. L. Fisher. Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1–18, January 1981.
- [46] W. Förstner. Fex website. <http://www.ipb.uni-bonn.de/ipb/projects/fex/fex.html>, Institute of Photogrammetry, University of Bonn, Germany.
- [47] W. Förstner. A framework for low level feature extraction. In J.O. Eklundh, editor, *Computer Vision - ECCV '94*, volume 801, pages 61–70. Springer-Verlag, 1994.
- [48] Robert M. Freund and Shinji Mizuno. Interior point methods: Current status and future directions. In H. Frenk et al., editor, *High Performance Optimization*, pages 441–466. Kluwer Academic Publishers, 2000.
- [49] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In Egon Balas and Jens Clausen, editors, *Integer Programming and Combinatorial Optimization*, volume 920, pages 1–13. Springer, 1995.
- [50] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1991.
- [51] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(6):721–741, Nov 1984.
- [52] A. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.
- [53] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer-Verlag New York Inc., May 2001.
- [54] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.
- [55] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Patt. Anal. Mach. Intell.*, 18(4):377–388, 1996.

- [56] A. Graham. *Kronecker Products and Matrix Calculus with Applications*. Ellis Horwood Limited and John Wiley and Sons, 1981.
- [57] V. Granville, M. Krivanek, and J.P. Rasson. Simulated annealing: A proof of convergence. *PAMI*, 16(6):652–656, June 1994.
- [58] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
- [59] S.W. Hadley, F. Rendl, and H. Wolkowicz. A new lower bound via projection for the quadratic assignment problem. *Math. of Operations Research*, 17:727–739, 1992.
- [60] P.M. Hahn. Tree elaboration strategies for branch-and-bound solution of the quadratic assignment problem. *Yugoslav Journal of Operations Research*, 11(1):41–60, 2001.
- [61] Edwin R. Hancock and Mario Vento, editors. *Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop, GbRPR 2003, York, UK, June 30 - July 2, 2003, Proceedings*, volume 2726 of *Lecture Notes in Computer Science*. Springer, 2003.
- [62] R. M. Haralick and L.G. Shapiro. The consistent labeling problem: part i. *IEEE Trans. Pattern Anal. Machine Intell.*, 1(2):173–184, 1979.
- [63] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pages 147–151, 1988.
- [64] William A. Harris, Jay P. Fillmore, and Donald R. Smith. Matrix exponentials—another approach. In *SIAM Review*, volume 43, pages 694–706. 2001.
- [65] H. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14:174–194, 1958.
- [66] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees. *Oper. Res*, 18:1138–1162, 1970.
- [67] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 1:6–25, 1971.
- [68] C. Helmberg, F. Rendl, and R. Weismantel. Quadratic knapsack relaxations using cutting planes and semidefinite programming. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science*, volume 1084, pages 175–189. Springer, 1996.
- [69] Christoph Helmberg. *Semidefinite Programming for Combinatorial Optimization*. PhD thesis, ZIB-Report 00-34, Habilitationsschrift, TU Berlin, Konrad-Zuse-Zentrum Berlin, October 2000.

- [70] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, (2):342–361, 1996.
- [71] L. Herault, R. Horaud, F. Veillon, and J.J. Neiz. Symbolic image matching by simulated annealing. *Proc. British Machine Vision Conference*, pages 319–324, 1990.
- [72] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.
- [73] J. H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor*, 1975.
- [74] J. Hopcroft and J. Wong. A linear time algorithm for isomorphism of planar graphs. In *Proceedings of the 6th ACM Symposium on Theory of Computing*, pages 172–184, 1974.
- [75] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Patt. Anal. Mach. Intell.*, 11(11):1168–1180, 1989.
- [76] ILOG. Cplex website. <http://www.ilog.com>.
- [77] S. Ishii and M. Sato. Doubly constrained network for combinatorial optimization. *Neurocomputing*, 2001. to appear.
- [78] M. Jerrum. Large cliques elude the metropolis process. *Random Structures Algorithms*, 3(4):347–359, 1992.
- [79] J.-M. Jolion and W. G. Kropatsch, editors. *Graph Based Representations in Pattern Recognition*. Springer-Verlag Wien New York, 1998.
- [80] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.
- [81] R. Karp. Reducibility among combinatorial problems. In R. Mille and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. New York: Plenum Press, 1972.
- [82] William Karush. Minima of functions of several variables with inequalities as side conditions. Master’s thesis, The University of Chicago, 1939.
- [83] J. Keuchel, C. Schnörr, C. Schellewald, and D. Cremers. Binary partitioning, perceptual grouping, and restoration with semidefinite programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1364–1379, Nov 2003.
- [84] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

- [85] Whoi-Yul Kim and Avinash C. Kak. 3-d object recognition using bipartite matching embedded in discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224–251, 1991.
- [86] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
- [87] J.V. Kittler and E.R. Hancock. Combining evidence in probabilistic relaxation. *IJPRAI*, 3(1):29–51, 1989.
- [88] V. Klee and G. L. Minty. How good is the simplex algorithm? *Inequalities III, Academic Press, New York*, pages 159–175, 1972.
- [89] M. Kojima, S. Kim, and H. Waki. A general framework for convex relaxation of polynomial optimization problems over cones. *Journal of Operations Research Society of Japan*, 46(2):125–144, 2003.
- [90] T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.
- [91] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics 21. Springer, Berlin Heidelberg New York, first edition 2000 edition, 2000.
- [92] M. Kočvara and M. Stingl. Pennon: a code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333, 2003.
- [93] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, University of California Press, Berkeley and Los Angeles.
- [94] C. Lemaréchal. Lagrangian relaxation. In M. Juenger and D. Naddef, editors, *Computational Combinatorial Optimization*, Lecture Notes in Computer Science 2241 (Springer Verlag, 2001), pages 115–160, 2001.
- [95] C. Lemaréchal and F. Oustry. Semidefinite relaxations and lagrangian duality with application to combinatorial optimization. Technical report, Institut National de Recherche en Informatique et en Automatique, INRIA, St Martin, France, 1999.
- [96] L. Lovász and A. Schrijver. Cones of matrices, and set functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, May 1991.
- [97] B. Luo and E.R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 23(10):1120–1136, 2001.
- [98] Joo Maciel and Joo Costeira. Towards a global solution of the correspondence problem. *pami*, Submitted to IEEE Trans.PAMI – to appear 2003.

- [99] A.W. Marshall and I. Olkin. *Inequalities: The Theory of Majorizations and Its Applications*. Academic Press, New York, NY, 1979.
- [100] S. Meshoul and M. Batouche. Robust point correspondence for image registration using optimization with extremal dynamics. In *DAGM02*, page 330 ff., 2002.
- [101] Bruno T. Messmer and H. Bunke. Subgraph isomorphism in polynomial time. Technical Report IAM 95-003, 1995.
- [102] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. Patt. Anal. Mach. Intell.*, 20(5):493–504, 1998.
- [103] M. Minoux. *Mathematical programming: theory and algorithms*. Wiley, 1986.
- [104] H. D. Mittelmann. An independent benchmarking of sdp and socp solvers. *Mathematical Programming Ser. B*, 95:407–430, 2003.
- [105] T.S. Motzkin and E.G. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian J. Math.*, 17:533–540, 1965.
- [106] K. G. Murty. *Linear complementarity, linear and nonlinear programming*. Heldermann Verlag, Berlin, 1988.
- [107] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.
- [108] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. J. Wiley, New York, 1988.
- [109] Y. Nesterov and A. Nemirovski. *Interior Point Polynomial Methods in Convex Programming*. SIAM Series In Applied Mathematics, Philadelphia., 1994.
- [110] Ivo Nowak. Locally exact lower bounds and optimality cuts for all-quadratic programs with convex constraints.
- [111] P. M. Pardalos and A. T. Phillips. A global optimization approach for solving the maximum clique problem. *Int. J. Computer Math.*, 33:209–216, 1990.
- [112] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is np-hard. *J. Global Optim.*, 1:15–22, 1991.
- [113] Panos M. Pardalos and Henry Wolkowicz, editors. *Quadratic Assignment and Related Problems*, volume 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1994.
- [114] M. Pelillo, K. Siddiqi, and S. Zucker. Continuous-based heuristics for graph and tree isomorphisms, 1999.

- [115] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for 0-1 quadratic programming. *Journal of Global Optimization*, (7):51–73, 1995.
- [116] B. Radig. Image sequence analysis using relational structures. *Pattern Recognition*, 17(1):161–167, 1984.
- [117] P. Raghavan and C. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
- [118] A. Rangarajan, A. Yuille, and E. Mjolsness. Convergence properties of the softassign quadratic assignment algorithm. 11(6):1455–1474, 1999.
- [119] Anand Rangarajan. Self annealing: Unifying deterministic annealing and relaxation labelling. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 229–244, 1997.
- [120] F. Rendl and R. Sotirov. Bounds for the quadratic assignment problem using the bundle method. Technical report, University of Klagenfurt, Austria, 2003.
- [121] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):420–433, 1976.
- [122] Stefan Roth. Analysis of a deterministic annealing method for graph matching and quadratic assignment problems in computer vision. Master’s thesis, CVGPR-group, University of Mannheim, May 2001.
- [123] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover’s distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
- [124] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association of Computing Machinery*, 23(3):555–565, 1976.
- [125] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transaction on Systems, Man and Cybernetics*, 13(3):353–362, 1983.
- [126] C. Schellewald, S. Roth, and C. Schnörr. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In B. Radig and S. Florczyk, editors, *Mustererkennung 2001*, volume 2191 of *Lect. Notes Comp. Science*, pages 361–368, Munich, Germany, Sept. 12–14 2001. Springer.
- [127] C. Schellewald and C. Schnörr. Subgraph matching with semidefinite programming. In Vito Di Gesù Alberto Del Lungo and Attila Kuba, editors, *Electronic Notes in Discrete Mathematics*, volume 12. Elsevier Science Publishers, 2003.

- [128] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc. New York, NY, USA, 1986.
- [129] R. Seidel. Constrained delaunay triangulations and voronoi diagrams with obstacles. In *Report 260 IIG-TU*, pages 178–191. Techn. Univ. Graz, Austria, 1988.
- [130] P.D. Seymour. Decomposition of regular matroids. *J. Combin. Theory Ser. B*, 28:305–359, 1980.
- [131] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *ICCV*, pages 222–229, 1998.
- [132] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals Mathematical Statistics*, 35(2):876–879, June 1964.
- [133] Martin Skutella. Convex quadratic programming relaxations for network scheduling problems. In *European Symposium on Algorithms*, pages 127–138, 1999.
- [134] Renata Sotirov. *Bundle Methods in Combinatorial Optimization*. PhD thesis, University of Klagenfurt, Austria, 2003.
- [135] E. Stiefel. Richtungsfelder und fernparallelismus in n- dimensionalem mannig faltigkeiten. *Commentarii Math. Helvetici*, 8:305–353, 1935–1936.
- [136] X. Sun, K. Mckinnon, and D. Li. A convexification method for a class of global optimization problems with application to reliability optimization. *Journal of Global Optimization*, 21(2):185–199, 2001.
- [137] Ph. L. Toint. On sparse and symmetric matrix updating subject to a linear equation. *Mathematics of Computation*, 31:954–961, 1977.
- [138] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.
- [139] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Patt. Anal. Mach. Intell.*, 10(5):695–703, 1988.
- [140] Lieven Vandenbergh and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.
- [141] D.L. Waltz. Understanding line drawings of scenes with shadows. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 19–92. McGraw-Hill, 1975.
- [142] Yuan-Kai Wang, Kuo-Chin Fan, and Jorng-Tzong Horng. Genetic-based search for error-correcting graph isomorphism. *IEEE TSMC: IEEE Transactions on Systems, Man, and Cybernetics*, 27, 1997.

-
- [143] R.C. Wilson and E.R. Hancock. Structural matching by discrete relaxation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 19(6), 1997.
 - [144] Gerhard Winkler. *Image analysis, random fields and Markov chain Monte Carlo methods : a mathematical introduction*. Berlin ; New York : Springer, 2003.
 - [145] Inc. Wolfram Research. *Mathematica*. Wolfram Research, Inc., Champaign, Illinois, 4.2 edition, 2002.
 - [146] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*, Boston, 2000. Kluwer Acad. Publ.
 - [147] Yinyu Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, 1997.
 - [148] X. Zhang and Y. Ye. *User's Guide of COPL-QP Computational Optimization Program Library: Convex Quadratic Programming*. University of Iowa, July 1998.
 - [149] Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *J. Combinat. Optimization*, 2(1):71–109, 1998.
 - [150] Quing Zhao. *Semidefinite Programming for Assignment and Partitioning Problems*. PhD thesis, University of Waterloo, 1996.